GRAHA ILMU

PEMROSESAN SINYAL DIGITAL

Nemuel Daniel Pah

PEMROSESAN
SINYAL DIGITAL

PEMROSESAN SINYAL DIGITAL

Nemuel Daniel Pah

GRAHA ILMU

PEMROSESAN SINYAL DIGITAL

oleh Nemuel Daniel Pah

GRAHA ILMU
Ruko Jambusari 7A Yogyakarta 55283
Telp: 0274-889398; Fax: 0274-889057; E-mail: info@grahailmu.co.id

Tajuk Entri Utama: Pah, Nemuel Daniel

PEMROSESAN SINYAL DIGITAL/Nemuel Daniel Pah
– Edisi Pertama. Cet. Ke-1. – Yogyakarta: Graha Ilmu, 2018
xvi + 238 hlm.; 24 cm

Bibliografi: 237
ISBN : 978-602-262-789-0
E-ISBN : 978-602-262-790-6

1. Elektronika Digital I. Judul

621.381

Prakata

Latar Belakang

Mata kuliah pemrosesan sinyal digital selalu dianggap sebagai mata kuliah yang sulit. Kesulitan ini bukan hanya menjadi pengalaman mahasiswa, tetapi juga dialami oleh dosen ketika harus mempersiapkan kuliah, bahkan ketika menjelaskan berbagai rumus di depan kelas. Kesulitan ini juga dialami oleh penulis ketika mendapat tugas untuk mengajar mata kuliah ini.

Ada sangat banyak buku teks tentang *Digital Signal Processing* yang membahas topik ini dengan sangat lengkap dan detail. Kelengkapan dan kedalaman dari buku-buku ini justru membuat kesulitan dalam mempersiapkan kuliah. Terlalu banyak bahan yang harus diajarkan, terlalu banyak rumus yang harus diturunkan dan dibuktikan, terlalu banyak konsep yang harus dipahami, terlalu banyak asumsi yang harus diterima, terlalu banyak simbol yang harus dihafal. Buku-buku ini cenderung membahas dari pendekatan sains, bukan pendekatan *engineering*. Kesulitan lainnya adalah karena adanya ketidakkonsistenan penggunaan simbol antara satu buku dengan buku lainnya. Kondisi inilah yang membuat pemrosesan sinyal digital terlihat sangat menyeramkan.

Setelah membaca berbagai buku, mencoret-coret di kertas, mencoba di komputer, menghitung dengan kalkulator,

menempuh berbagai kebingungan, berdiskusi dengan rekan dosen, akhirnya mata kuliah ini berhasil saya ajarkan dengan hasil yang memuaskan. Mahasiswa dapat mengerti dan menjadi tertarik untuk mempelajari bidang ini lebih lanjut. Pengalaman dan materi yang diajarkan pada setiap minggu perkuliahan ini menjadi begitu berharga bagi saya sehingga didokumentasikan dalam bentuk buku ajar kuliah ini.

Motivasi

Buku ini ditulis karena dorongan untuk membagikan pengalaman dan memberi kuliah ini agar rekan-rekan dosen yang lain tidak perlu menempuh kesulitan yang tidak perlu dalam mempersiapkan kuliah pemrosesan sinyal digital. Buku ini ditulis agar ada suatu set bahan ajar yang siap disajikan sehingga, baik dosen maupun mahasiswa, dapat menyajikan kuliah dan mempelajari bidang pemrosesan sinyal digital ini dengan kualitas konsep yang baik, kedalaman yang cukup, dan contoh-contoh perhitungan yang lengkap.

Bab-bab dalam buku ini dibuat dengan konsep yang *compact*. Setiap bab dibuat tidak terlalu panjang sehingga suatu konsep dapat ditanamkan dalam maksimum satu kali perkuliahan 3 sks. Bab-bab ini juga berisi contoh perhitungan dan soal latihan untuk menajamkan pemahaman konsep dan *skill* engineering.

Sekalipun materi dalam buku ini telah disiapkan dengan sangat baik dan siap disajikan dalam perkuliahan, buku ini tidak ditujukan untuk menggantikan peran dari buku-buku teks utama dalam bidang *Digital Signal Processing*. Bab-bab dalam buku ini hanya mencapai tingkat kedalaman untuk sesi perkuliahan. Rumus-rumus dan prosedur perhitungan yang digunakan hanya dipaparkan dan dijelaskan secara konseptual tetapi tidak dirinci penurunanya secara matematis.

Bab-bab ini lebih menuntut mahasiswa untuk memahami, menerima dan memakai prosedur dan rumus yang ada untuk analisis dan desain sistem pemrosesan sinyal digital. Setelah memahami materi dalam buku ini, dosen pengasuh dan mahasiswa sangat dianjurkan untuk menggali lebih dalam melalui buku-buku teks. Hal ini terlebih lagi ditekankan bagi para dosen. Tidak cukup bagi dosen untuk hanya mempersiapkan perkuliahan dari buku ini saja. Buku ini hanya digunakan sebagai media komunikasi antara dosen dan mahasiswa disajikan di kelas. Konsep dan kedalaman pemahaman harus digali dari berbagai buku.

Capaian Pembelajaran

Buku Pemrosesan Sinyal Digital ini merupakan buku ajar yang dapat digunakan sebagai bahan ajar maupun bahan belajar dari dosen dan mahasiswa untuk mata kuliah program sarjana. Materi dalam buku ini dirancang untuk suatu mata kuliah dengan beban 3 sks.

Setelah menyelesaikan pembelajaran dengan buku ini maka mahasiswa diharapkan akan dapat:

- Menjelaskan konsep sinyal dan sistem serta mendeskripsikan sinyal dalam persamaan matematis baik kontinyu maupun diskrit.
- Mentransformasi sinyal dari domain waktu ke domain frekuensi dan sebaliknya
- Menganalisa sinyal berdasarkan spektrum frekuensinya.
- Merancang dan mengimplementasi filter digital.
- Merancang sistem pemrosesan sinyal untuk aplikasi sinyal suara dan dapat mengimplementasikan dalam bentuk program aplikasi komputer.

Struktur Buku

Buku ini terdiri atas 17 bab yang dapat dibagi menjadi dua bagian. Bagian pertama adalah pemahaman tentang konsep pemrosesan sinyal dan analisis sinyal. Bagian ini bertujuan untuk memberikan pemahaman tentang sinyal, baik dalam domain waktu maupun dalam domain frekuensi, serta memberikan pemahaman tentang konsep dan prosedur untuk menganalisis sinyal berdasarkan spektrum frekuensinya. Bagian yang pertama ini terdiri dari 7 bab yaitu:

- Bab 1: Pengantar Pemrosesan Sinyal, yang berisi konsep pemrosesan sinyal dan petunjuk dasar penggunaan Matlab.
- Bab 2: Domain Waktu dan Domain Frekuensi, yang memberian fondasi dasar tentang pemahaman sinyal dan sistem dalam kedua domain.
- Bab 3: Sinyal Diskrit, memberikan langkah pertama untuk beralih dari sinyal analog ke sinyal diskrit.
- Bab 4: Discrete Fourier Transform, yang mengajarkan tentang teori dan prosedur perhitungan transformasi untuk beralih dari domain waktu ke domain frekuensi dan sebaliknya secara diskrit.
- Bab 5: Spektrum Frekuensi bertujuan untuk memberikan pemahaman dan skill untuk menganalisis spektrum frekuensi dari sinyal.
- Bab 6: Spektrogram bertujuan untuk memberikan pemahaman dan skill untuk menganalisis spektrum frekuensi dari sinyal non stasioner.
- Bab 7: Analisis Sinyal Suara yang merupakan bab aplikasi dari teori yang sudah dipelajari pada Bab 1 sampai Bab 6. Bab ini disajikan dalam bentuk bahan praktikum.

Bagian kedua terdiri dari 10 bab yang membahas secara spesifik tentang sistem diskrit. Pembahasan sistem diskrit pada bab-bab ini diarahkan pada pemahaman cara kerja

sistem diskrit dan berbagai metode untuk mendesain sistem diskrit berupa filter digital IIR dan FIR. Bab-bab ini adalah:

- Bab 8: Sitem Diskrit yang memberikan pemahaman tentang cara kerja sistem diskrit yang dinyatakan dengan impulse response dan sistem diskrit yang dinyatakan dalam bentuk *transfer function*.
- Bab 9: Transformasi Z, yang menajdi fondasi untuk merelasikan sistem dan sinyal analog dengan sistem dan sinyal diskrit.
- Bab 10: Konsep Filter, menjadi pintu masuk untuk memahami konsep filter.
- Bab 11: Filter Digital, memberikan gambaran detail tentang berbagai format dan bentuk implementasi dari filter digital.
- Bab 12 sampai Bab 14 berisi prosedur untuk desain filter FIR dengan metode sampling frekuensi, metode window, dan metode optimal.
- Bab 15 sampai Bab 17 berisi prosedur untuk mendesain filter IIR dengan metode *pole-zero placement*, transformasi bilinier, dan metode *impulse invariant*.

Beberapa bagian dalam buku ini dapat diaplikasikan dengan menjalankan perintah dari program komputer Matlab atau Octave. Program-program tersebut beserta dengan file data dapat diperoleh pada situs web buku ini dengan menghubungi dengan mengirim email kepada penulis di nemuelpah@staff.ubaya.ac.id.

Saran Penggunaan

Bab-bab dalam buku ini dirancang untuk perkuliahan pemrosesan sinyal digital sebanyak 16 minggu pertemuan/evaluasi, masing masing 3 sks (3 sesi tatap muka @ 50 menit). Berdasarkan rancangan ini maka disarankan untuk menggunakan bab-bab ini dengan urutan sebagai berikut:

Minggu 1: Bab 1
Minggu 2: Bab 2
Minggu 3: Bab 3
Minggu 4: Bab 4
Minggu 5: Bab 5
Minggu 6: Bab 6
Minggu 7: Bab 7
Minggu 8: Evaluasi (UTS)
Minggu 9: Bab 8
Minggu 10: Bab 9
Minggu 11: Bab 10
Minggu 12: Bab 12
Minggu 13: Bab 13 dan Bab 14
Minggu 14: Bab 15 dan Bab 16
Minggu 15: Bab 17
Minggu 16: Evaluasi (UAS)

Pembagian ini hanya merupakan saran saja. Dosen pengajar dapat menggunakan rancangan yang lain dan disesuaikan dengan kecepatan mahasiswa dalam memahami isi dari setiap bab.

Penghargaan

Materi dalam buku ini lahir dari proses perkuliahan di Jurusan Teknik Elektro, Universitas Surabaya pada mata kuliah *Signal Processing* (61A262) mulai tahun ajaran 2008/2009 sampai tahun ajaran 2017/2018 ketika telah berubah nama menjadi mata kuliah Pemrosesan Sinyal Digital (1601A042). Kesempurnaan buku ini hanya bisa terjadi karena peran yang sangat aktif dari semua mahasiswa untuk menggunakan, mencoba, dan memberi kritik serta ide perbaikan. Saya sangat bangga dan berterima kasih atas semua mahasiswa saya yang telah menempuh mata kuliah ini dan atas prestasi yang telah mereka capai.

Penulis menyampaikan terima kasih kepada Djuwari, Ph.D. yang dengan tidak letih-letihnya terlibat dalam berbagai diskusi yang mendalam dengan penulis untuk memahami berbagai konsep dan prosedur dari pemrosesan sinyal digital. Tanpa diskusi-diskusi ini, banyak konsep yang belum dapat dipahami secara tuntas oleh penulis. Beliau juga sangat teliti dan kritis membaca, mencoba dan memberikan berbagai masukan serta ide pengembangan buku ini.

Secara khusus penulis menyampaikan terima kasih dan penghargaan yang sebesar-besarnya kepada Direktorat Jenderal Penguatan Riset dan Pengembangan pada Kementrian Riset, Teknologi, dan Pendidikan Tinggi Republik Indonesia yang telah memberikan Hibah Penerbitan Buku Ajar 2018 kepada penulis untuk menerbitkan buku ini.

Kami juga menyampaikan terima kasih kepada Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng. yang telah memberikan dorongan dan bimbingan dalam penulisan dan penerbitan buku ini.

Akhirnya, saya berterima kasih kepada istri, Vivi, dan kedua anak yang tercinta, Heidi dan Daniel atas dorongan dan waktu yang terus menerus diberikan agar dapat menulis buku ini sampai selesai.

Nemuel Daniel Pah
Surabaya, Oktober 2018

Daftar Isi

1

Pengantar Pemrosesan Sinyal

Pemrosesan sinyal adalah suatu bidang ilmu yang mengembangkan dan mengaplikasikan teori-teori matematika dan pemrograman komputer untuk memproses atau memanipulasi sinyal agar memiliki kegunaan nilai yang lebih tinggi. Teori-teori pemrosesan sinyal dapat diimplementasikan dalam bentuk perangkat keras maupun perangkat lunak.

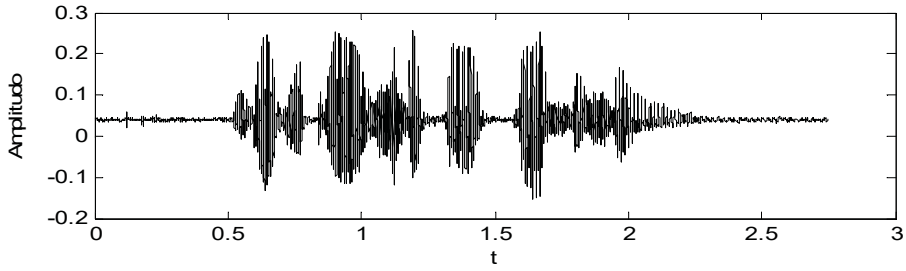
Bab ini bertujuan untuk memberikan gambaran secara menyeluruh tentang bidang pemrosesan sinyal, baik tentang definisi sinyal dan pemrosesan sinyal, aplikasinya, serta teknik-teknik dasar menggunakan perangkat lunak untuk implementasi. Setelah mempelajari materi dalam bab ini maka mahasiswa akan dapat menjelaskan definisi pemrosesan sinyal digital, mengidentifikasi bentuk aplikasi dari pemrosesan sinyal digital dan menggunakan perangkat lunak untuk mengimplementasikan algoritma pemrosesan sinyal digital.

1.1 Sinyal

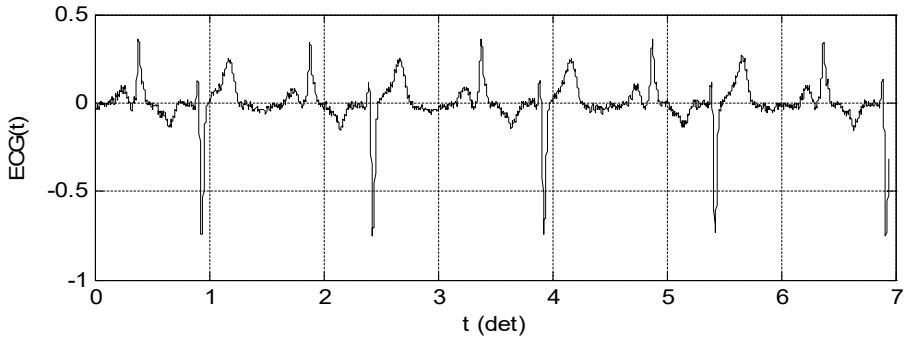
Sinyal (‘***signal***’ dalam bahasa Inggris) berasal dari kata ‘sign’ yang berarti tanda atau isyarat. Dalam pemrosesan sinyal, kata *‘sinyal’* diartikan sebagai fungsi yang mengandung informasi. Karena merupakan fungsi maka

sinyal memenuhi semua persyaratan matematis sebagai sebuah fungsi, tetapi tidak semua fungsi matematis memiliki kandungan informasi di dalamnya.

Sinyal suara adalah perubahan tekanan udara sebagai fungsi waktu yang mengandung arti/informasi sesuai pesan dari kata-kata yang diucapkan. Sinyal ECG (elektrocardiograph) adalah fungsi tegangan listrik terhadap waktu yang direkam dari aktifitas jantung. Sinyal ini mengandung informasi tentang kondisi kesehatan jantung. Gambar 1.1 menunjukkan sebuah sinyal ucapan '*digital signal processing*'. Sinyal ini tersimpan dalam file `sinyal1_1.mat` yang dapat diunduh dari situs buku ini. Gambar 1.2 menunjukkan sinyal ECG (Electrocardiography).



Gambar 1.1
Sinyal dari ucapan 'digital signal processing'



Gambar 1.2
Sinyal Electrocardiography

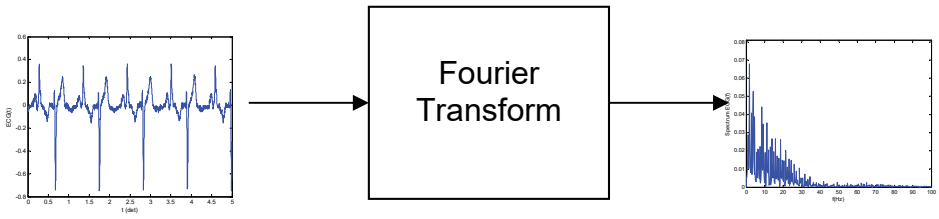
1.2 Pemrosesan

Istilah "proses" dalam kata pemrosesan sinyal adalah semua tindakan memanipulasi sebuah sinyal secara matematis untuk tujuan:

- **Analisa:** menggali informasi yang terkandung dalam suatu sinyal.
- **Kompresi:** memampatkan informasi dalam suatu sinyal sehingga hanya memerlukan ruang sekecil mungkin untuk disimpan atau ditransmisikan.
- **Pengayaan:** mengubah suatu sinyal ke dalam bentuk yang lain sehingga memiliki nilai yang lebih tinggi (baik nilai kandungan informasi maupun nilai kejelasan informasi)
- **Sintesa:** menggunakan berbagai teknik matematis untuk menghasilkan sinyal dengan kualitas dan kandungan informasi tertentu.

Ada banyak proses yang dapat digunakan dalam pemrosesan sinyal namun pembahasan dalam buku ini hanya akan dibatasi pada pemrosesan sinyal untuk *analisis spektrum* dan *filter*. Analisis spektrum dan filter merupakan pemrosesan paling dasar pada bidang pemrosesan sinyal. Gambar 1.3 menunjukan diagram blok dari suatu filter sedangkan Gambar 1.4 menunjukan diagram blok dari analisis spektrum.

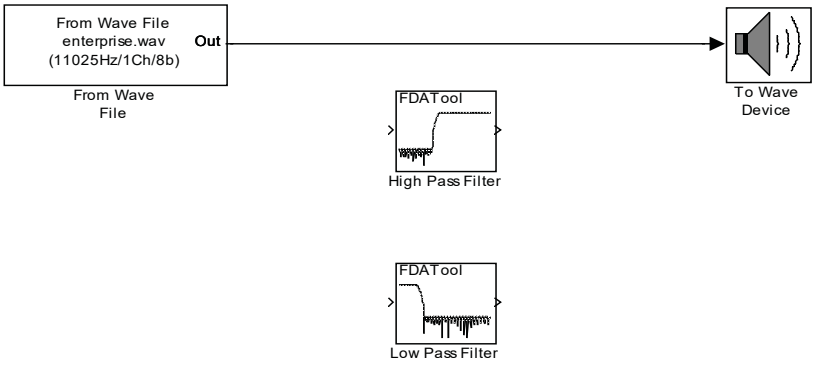
Gambar 1.3
Diagram Blok dari Filter



Gambar 1.4
Diagram Blok dari Analisis Spektrum

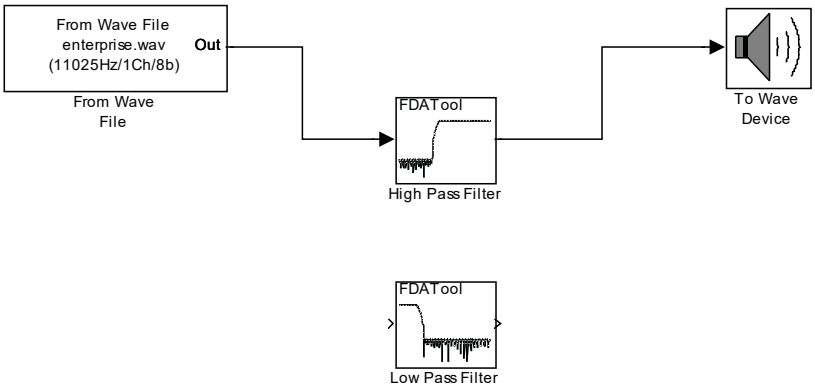
1.3 Sistem Pemrosesan Sinyal

Untuk mengenal lebih dalam tentang pemrosesan sinyal, bukalah file `sim1_1.mdl` (dapat diunduh pada situs buku) dengan menggunakan software Simulink. Filter ini berisi simulasi sistem pemrosesan sinyal untuk memisahkan bunyi peluit dari ucapan manusia. Sistem ini terdiri dari rekaman `enterprise.wav`, dua buah filter dan loudspeaker. Pertama-tama file wav tersebut langsung dihubungkan ke loudspeaker seperti pada Gambar 1.5.



Gambar 1.5
Sistem pemrosesan sinyal pada sim1_1.mdl

Jalankan sistem tersebut (klik pada menu `simulation -> start`) dan dengarlah suara/bunyi yang ada dalam file tersebut. File tersebut berisi siulan dan suara orang. Hubungkanlah file wav tersebut melalui filter highpass seperti pada Gambar 1.5. Jalankan sistem tersebut dan perhatikan efek yang dikerjakan oleh filter tersebut. Filter tersebut meredam suara orang pada rekaman itu. Lakukan hal yang sama dengan filter lowpass dan perhatikan kegunaan dari filter-filter tersebut.



Gambar 1.6
Menggunakan High Pass Filter

Anda dapat menggunakan dan memodifikasi filter-filter tersebut untuk bereksperimen dengan sistem pemrosesan sinyal ini. Pada situs buku ini terdapat pula file `sim1_2.mdl`, `sim1_3.mdl`, dan `sim1_4.mdl`. Bukalah file-file tersebut dan lakukan eksperimen atau modifikasi untuk mengenal lebih dalam tentang berbagai sistem pemrosesan sinyal.

Semua sistem dalam bab-bab ini adalah sistem yang bersifat *causal* dan *linear time invariant* (LTI). Suatu sistem bersifat *causal* jika sistem tersebut hanya dapat mengeluarkan output

berdasarkan input yang sudah diberikan kepadanya. Sistem ini tidak dapat melakukan ramalan yaitu menghasilkan output sebelum menerima input.

Sistem LTI adalah sistem yang bersifat linier, dan memiliki properti (sifat) yang tidak berubah terhadap waktu. Penjelasan detail dari sifat-sifat ini dapat dilihat dalam berbagai buku tentang sinyal dan sistem.

1.4 Aplikasi Pemrosesan Sinyal

Pemrosesan sinyal dapat diaplikasikan dalam berbagai bidang antara lain:

- Audio (Sistem rekaman audio, kompresi audio, midi, audio enhancement, equaliser, filter, noise cancelation, echo, 3D sound, dan transmisi audio)
- Speech (speech recognition, speech sythesis, text to speach, speaker recognition, digital audio)
- Sonar dan radar (aplikasi militer, pesawat, kapal selam)
- Sensor array
- Spectral estimation
- Image processing
- Communication (DTMF, transmisi data)
- Biomedical (Ultrasound, EEG, ECG, EMG)
- Seismic data (pertambangan, gempa bumi, pengaruh astronomi)

1.5 Software Pemrosesan Sinyal

Pemrosesan sinyal dapat diimplementasi dan dipelajari secara analitis maupun dengan menggunakan berbagai software. Secara garis besar, software untuk pemrosesan sinyal dapat dibagi atas dua kategori yaitu software simulasi dan software untuk implementasi ke hardware.

Pada bab ini akan digunakan Matlab sebagai software simulasi sedangkan untuk implementasi hardware dapat digunakan software penunjang untuk DSP Processor TMS. Masih banyak software simulasi yang dapat digunakan seperti Scilab, Octave, R-Lab, maupun bahasa pemrograman murni seperti C++. Untuk beberapa perhitungan akan juga digunakan software spreadsheet.

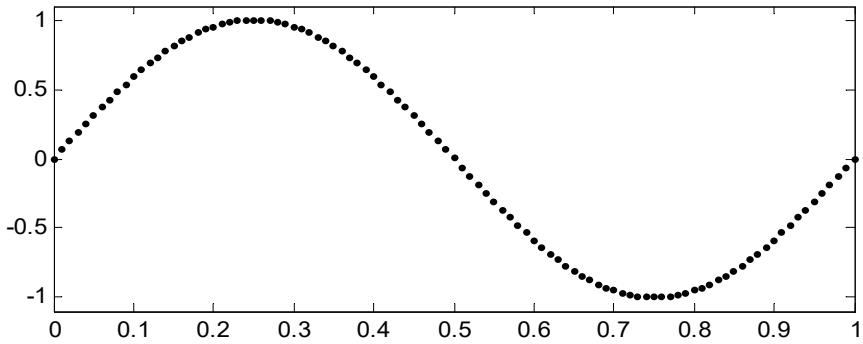
1.6 Pemrosesan Sinyal dengan MATLAB™

Bab-bab dalam buku ini berasumsi bahwa anda telah dapat menggunakan Matlab pada level pemula. Penjelasan pada bagian ini akan lebih menekankan pada penggunaan Matlab dalam pemrosesan sinyal. Bagi anda yang belum pernah menggunakan Matlab, sangat dianjurkan untuk mempelajari penggunaan dasar Matlab dengan menggunakan buku-buku pengantar Matlab maupun berbagai tutorial yang ada di website. Langkah terpenting yang harus anda lakukan selanjutnya adalah mencoba menggunakan Matlab, mencoba bereksperimen, membaca 'help' dan mencoba lagi. Berikut ini adalah beberapa perintah Matlab yang penting digunakan pada pemrosesan sinyal.

Selain Matlab, anda dapat juga menggunakan software lainnya yang mirip dengan Matlab seperti Octave dan Scilab. Hampir semua instruksi Matlab dapat dijalankan dengan Octave maupun Scilab.

Membuat dan Menggambar Sinyal

Matlab adalah software yang bekerja secara digital sehingga sebuah sinyal hanya bisa dibuat dalam bentuk digital yaitu kumpulan dari banyak titik seperti pada Gambar 1.7.



Gambar 1.7
Kumpulan titik yang membentuk kurva sinyal

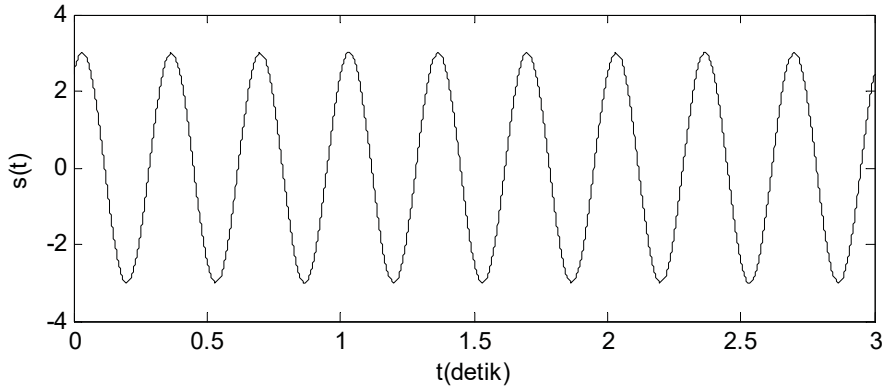
Untuk membuat sinyal di Matlab, kita harus mendefinisikan titik-titik tersebut dengan cara mendefinisikan waktu, t, dan sinyal, s(t). Perintah berikut ini digunakan untuk membuat dan menggambar sinyal

$$s(t) = 3 \cos (2 \pi 3 t - 0.2\pi)$$

untuk t dari 0 sampai 3 detik.

```
>> t=0:0.001:3;
>> s=3*cos(2*pi*3*t-0.2*pi);
>> plot(t,s);
>> axis([0 3 -4 4]);
>> xlabel('t(detik)');
>> ylabel('s(t)');
```

Perintah-perintah tersebut akan menghasilkan gambar berikut ini.



Gambar 1.8
Sinyal yang digambar dengan Matlab

Angka 0.001 pada definisi *t* menentukan jarak antara titik. Jika angka ini semakin kecil maka jarak antara titik akan semakin dekat. Jarak ini dikenal dengan istilah 'periode sampling' yang akan dipelajari lebih lanjut di Bab 3. Cobalah berkesperimen dengan perintah-perintah di atas dan bacalah 'help' dari perintah-perintah itu untuk memahaminya.

Jika yang diinginkan adalah gambar dalam bentuk kumpulan titik (seperti pada Gambar 1.7) maka gantilah perintah plot di atas dengan:

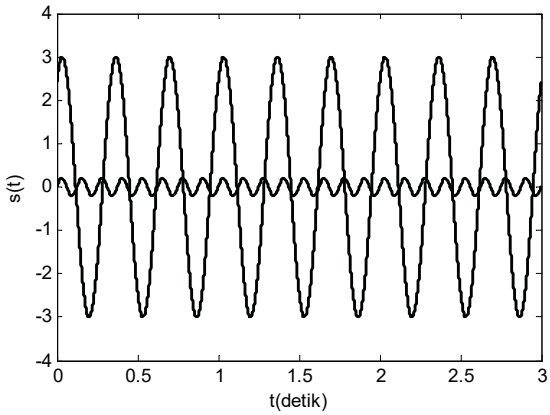
```
>> plot(t,s,'.');
```

Jika anda ingin menggambarkan dua buah sinyal dalam axis yang sama maka gunakanlah perintah 'hold on' dan 'hold off' seperti contoh berikut:

```
>> t=0:0.001:3;
>> s=3*cos(2*pi*3*t-0.2*pi);
>> plot(t,s);
>> axis([0 3 -4 4]);
>> xlabel('t(detik)');
>> ylabel('s(t)');
```

```
>> hold on

>> s2=0.2*sin(2*pi*8*t);
>> plot(t,s2);

>> hold off
```

Perintah di Matlab di atas akan menghasilkan plot seperti pada Gambar 1.9.



Gambar 1.9
Dua buah sinyal yang digambar dengan perintah *'hold'*

Menjumlahkan Sinyal

Jika anda ingin menjumlahkan beberapa buah sinyal (misalnya menjumlahkan sinyal s dan s2 di atas) maka gunakan perintah:

```
>> sTotal=s+s2;
```

Hal yang perlu diingat adalah banwa **Matlab bersifat 'case sensitive'** sehingga membedakan huruf besar dan huruf kecil

Sebelum menggunakan perintah ini, pastikan kedua sinyal tersebut memiliki format dan jumlah titik yang sama. Anda dapat memeriksanya di window 'workspace' atau menggunakan perintah

```
>> size(s)
ans =
        1       3001
>> size(s2)
ans =
        1       3001
```

Jika sinyal anda memiliki panjang (jumlah titik) atau format yang berbeda maka anda dapat memotong sinyal tersebut atau men-transpose. Untuk memotong sinyal s sebanyak 1000 titik:

```
>> s=s(1:1000);
```

Untuk men-transpose sinyal s gunakan

```
>> s=s';
```

Untuk menyimpan satu atau lebih variabel (misalnya *s* dan *t*) gunakan perintah:

```
>> save namafile s t
```

Untuk membaca variabel dalam file tersebut gunakan perintah:

```
>> load namafile
```

Mengakses HELP

Selain perintah-perintah di atas, masih sangat banyak perintah Matab yang harus dikenali dan digunakan. Untuk menggunakan bantuan help di Matlab, ketik:

```
>> help
```

atau

```
>> helpwin
```

Jika anda mencari bantuan untuk perintah tertentu (misalnya untuk perintah 'plot') maka ketik:

```
>> help plot
```

Untuk bantuan perintah 'save', ketik:

```
>> help save
```

Jika anda memerlukan bantuan tetapi anda tidak tahu nama perintah maka gunakan 'lookfor'. Misalnya anda ingin mendapat bantuan tentang label. Ketika anda ketik

```
>> help label
```

Matlab tidak menemukan perintah 'label', maka ketiklah dalam:

```
>> lookfor label
```

Matlab akan memberitahu semua perintah yang berkaitan dengan label. Catatan: Matlab membutuhkan waktu yang lama untuk perintah 'lookfor'.

Jika anda benar-benar bingung, anda dapat menggunakan perintah 'why' dengan mengetikkan:

```
>> why
```

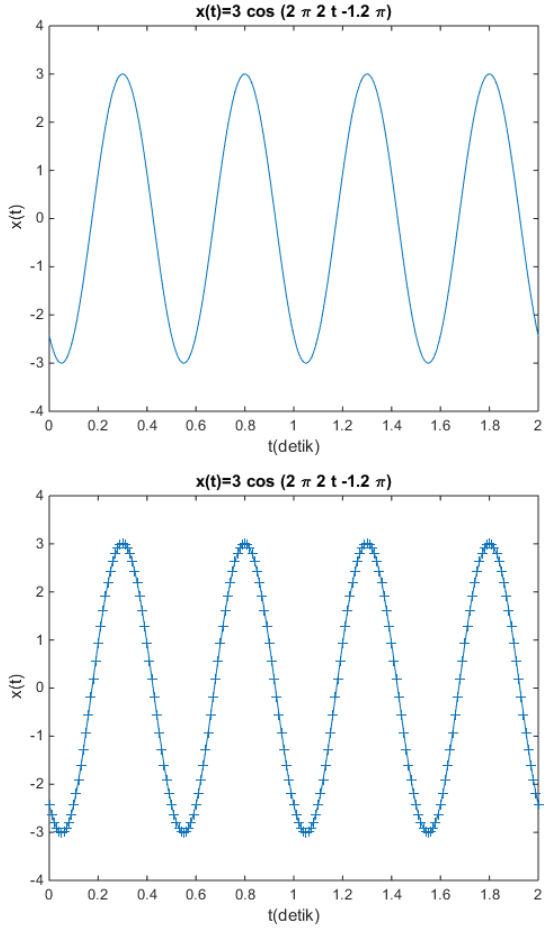
Dengan mengacu pada berbagai tutorial dan help dari Matlab, kerjakanlah soal latihan berikut ini untuk membiasakan anda dengan penggunaan Matlab.

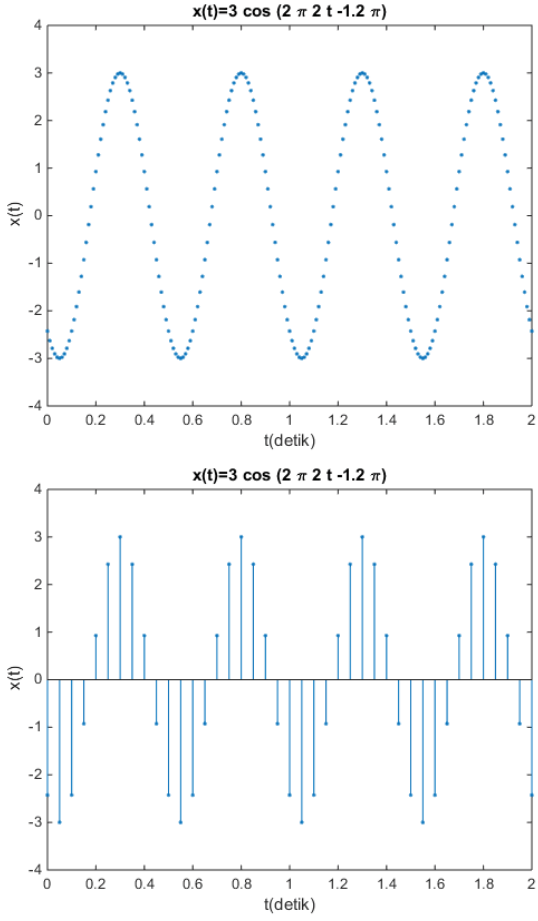
SOAL LATIHAN

1. Gunakan Matlab untuk membuat gambar sinyal

$$x(t) = 3 \cos (2 \pi 2 t - 1.2\pi)$$

seperti pada gambar-gambar berikut ini:

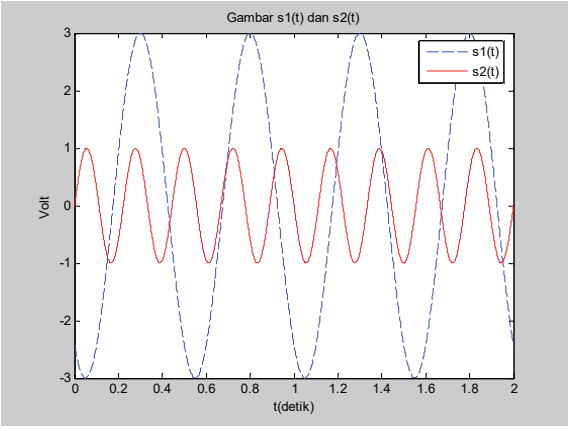




2. Gunakan Matlab untuk membuat gambar sinyal

   $s1(t) = 3 \cos (2 \pi 2 t - 1.2\pi)$ dan $s2(t) = 1 \sin (2 \pi 4.5 t)$

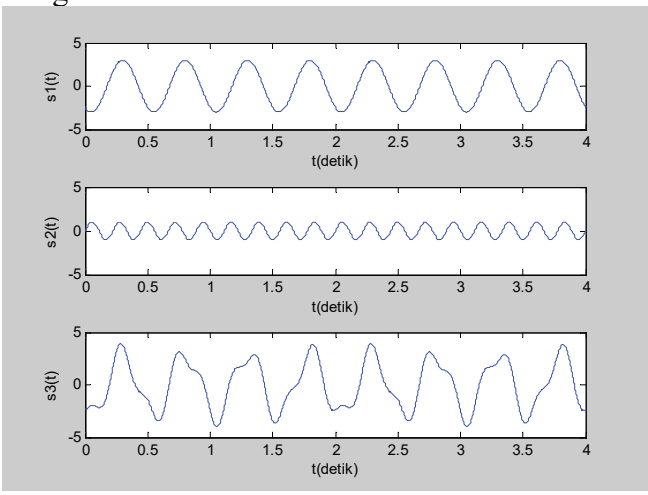
   seperti pada gambar berikut ini:



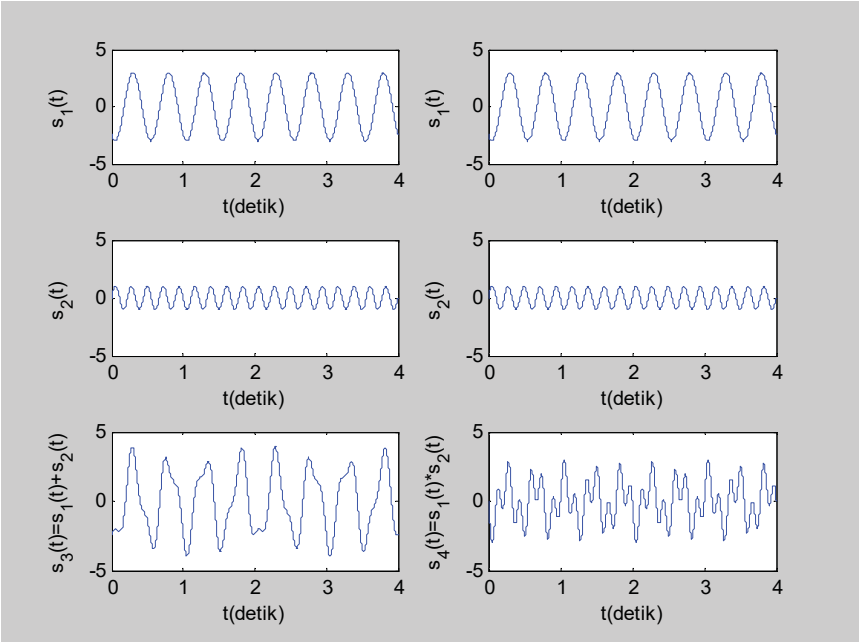
3. Gunakan Matlab untuk membuat gambar sinyal

$s1(t) = 3 \cos (2 \pi 2 t - 1.2\pi)$,
$s2(t) = 1 \sin (2 \pi 4.5 t)$ dan
$s3(t) = s1(t) + s2(t)$

seperti pada gambar berikut ini:



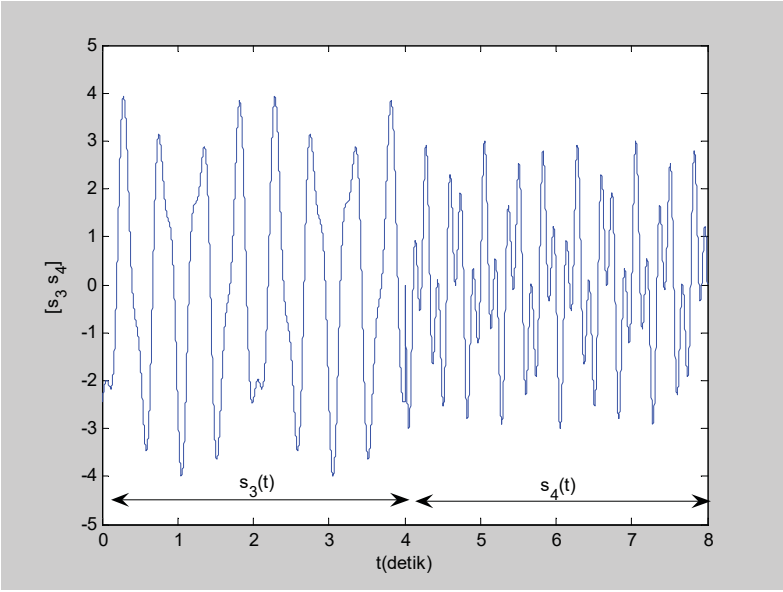
4. Buatlah file `*.m` untuk melakukan soal nomor 3 di atas. Lakukan modifikasi pada file `*.m` tersebut agar menampilkan gambar seperti berikut ini:



5. Simpanlah variabel *t* dan *s3* dalam sebuah file.

6. Gabungkan variabel *t* dan *s4* dalam sebuah variabel, *s*, dimana *t* menempati kolom pertama dan *s4* menempati kolom kedua. Simpanlah variabel *s* dalam sebuah file.

7. Hapuslah semua variabel dari memory Matlab dengan perintah 'clear all', kemudian bacalah kedua file yang dibuat di soal nomor 5 dan 6. Sambunglah sinyal *s3* dan *s4* kemudian gambarlah dengan perintah 'plot' sehingga tampil seperti gambar berikut ini:


[s3 s4]
t(detik)
s3(t)
s4(t)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
1
2
3
4
5
6
7
8

2

Domain Waktu dan Domain Frekuensi

Bab ini akan memberikan pemahaman secara mendasar tentang konsep penting dalam mengerti pemrosesan sinyal yaitu konsep tentang sinyal dan representasi sinyal dalam domain waktu dan domain frekuensi. Bab ini merupakan bagian yang sangat penting karena menjadi fondasi pemahaman dari semua aspek dalam teori pemrosesan sinyal. Jika kita gagal mengerti tentang adanya dua domain dan gagal mengerti relasi antara kedua domain ini maka kita akan kehilangan semua kerangka dasar pemrosesan sinyal.

Setelah mempelajari materi dalam bab ini maka mahasiswa akan dapat menggambar dan menuliskan persamaan sinyal dalam domain waktu dan domain frekuensi, serta dapat mengidentifikasi relasi antara sinyal dalam domain waktu dan domain frekuensi.

2.1 Sinyal Sinusoidal

Salah satu bentuk sinyal yang paling sering digunakan dalam pemrosesan sinyal adalah sinyal sinusiodal. Ekspresi

matematis dari sinyal sinusiodal, $x(t)$, selalu ditulis dalam bentuk kosinus sebagai berikut:

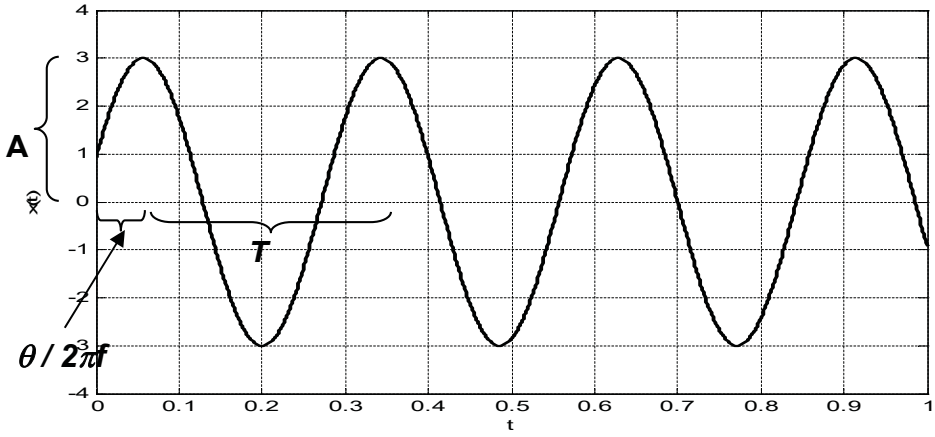
$$x(t) = A \cos (2 \pi f t + \theta) \quad \text{atau}$$
$$x(t) = A \cos (\omega t + \theta)$$

dimana:

t	adalah waktu (detik)
A	adalah amplitudo (berupa tegangan, arus, atau besaran amplitodu lainnya)
ω	adalah frekuensi angular (rad/s); $\omega = 2 \pi f$
f	adalah frekuensi (Hz); $f = 1/T$ (T adalah periode)
θ	adalah fase (derajat atau rad). Jika θ tidak nol maka sinyal akan bergeser sebesar θ/ω detik.

Contoh sinyal sinusoidal

$$x(t) = A \cos (2 \pi f t - \theta)$$



Gambar 2.1
Sinyal Sinusiodal

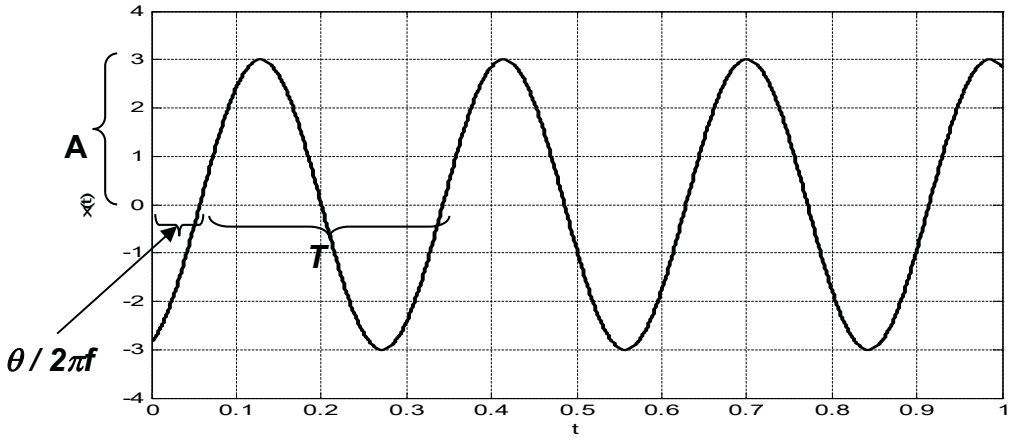
Catatan penting:
Dalam pemrosesan sinyal, ekspresi matematis dari sinyal sinusoidal adalah dalam bentuk kosinus,

$$x(t) = A \cos (2 \pi f t + \theta)$$

Ekspresi matematis sinyal sinusoidal dapat juga ditulis dalam bentuk *'sin'* sebagai berikut:

$$x(t) = A \sin (2 \pi f t - \theta)$$

Sinyal sinusoidal adalah sinyal kosinus yang digeser -90 derajat. Gambar dari sinyal sinusiodal di atas adalah seperti pada Gambar 2.2.



Gambar 2.2
Sinyal Sinusoidal $x(t) = A \sin (2 \pi f t - \theta)$

2.2 Analisis Fourier

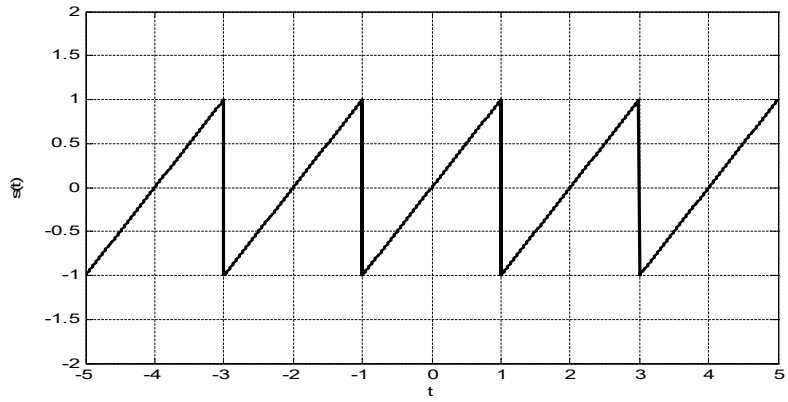
Gambar 2.3
Jean Baptiste Joseph Fourier (1768 -1830)
(sumber https://en.wikipedia.org/wiki/Joseph_Fourier)

Salah satu kontribusi besar dari Joseph Fourier adalah 'analisis Fourier' yang menyatakan bahwa:

*"setiap fungsi (termasuk **sinyal**) adalah merupakan penjumlahan dari satu atau banyak sinusoidal dengan berbagai amplitudo, frekuensi, dan fase"*

Sebagai contoh, sebuah sinyal gigi gergaji:

$$s(t) = \begin{cases} t & untuk \quad -1 < t < 1 \\ s(t-2) & untuk \quad t \ lain \end{cases}$$

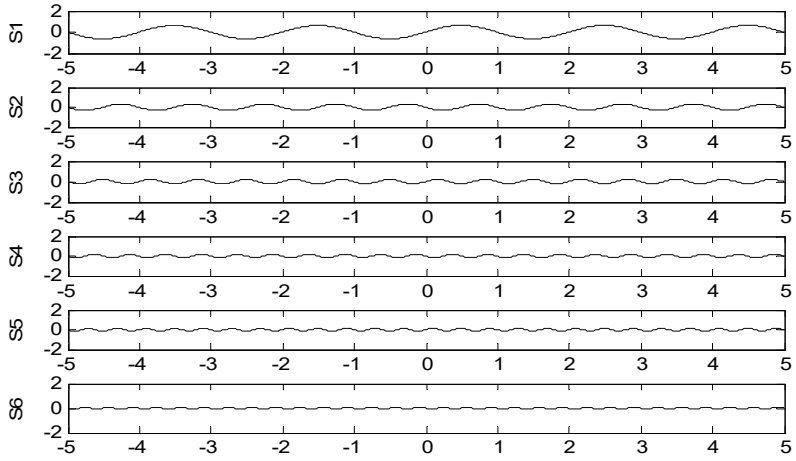


Gambar 2.4
Sebuah sinyal gigi gergaji, *s(t)*

dapat dibentuk dari penjumlahan banyak sinusiodal:

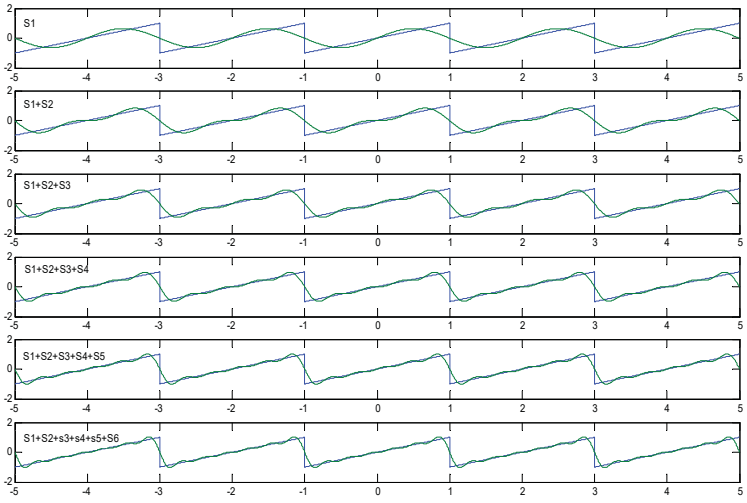
$$s(t) = 0.6366\cos(2\pi 0.5t - 1.57) - 0.3183\cos(2\pi 1t - 1.57) + 0.2122\cos(2\pi 1.5t - 1.57) - 0.1592\cos(2\pi 2t - 1.57) + 0.1273\cos(2\pi 2.5t - 1.57) - 0.1061\cos(2\pi 3t - 1.57) + \ldots$$

Gambar dari ke-enam sinusoidal adalah:



Gambar 2.5
Komponen sinusoidal sinyal *s(t)*

Jika sinyal sinusoidal tersebut dijumlahkan maka akan menghasilkan sinyal *s(t)* seperti terlihat pada Gambar 2.6.



Gambar 2.6
Penjumlahan komponen sinusoidal sinyal s(t)

Semakin banyak suku sinusoidal yang dijumlahkan maka akan semakin menyerupai s(t).

Dengan kata lain:

*setiap fungsi (termasuk **sinyal**) dibentuk oleh satu atau banyak sinusoidal dengan berbagai amplitudo, frekuensi, dan fase,*

atau

*setiap fungsi (termasuk **sinyal**) dapat diuraikan menjadi satu atau banyak sinusoidal dengan berbagai amplitudo, frekuensi, dan fase.*

Fungsi-fungsi sinusoidal yang membentuk suatu sinyal disebut **komponen sinusoidal dari suatu sinyal**.

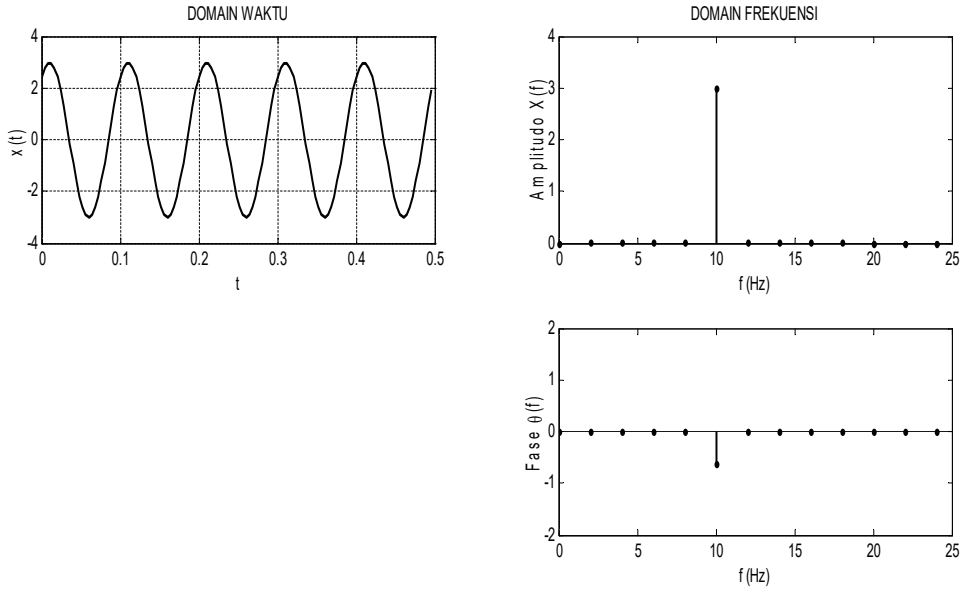
2.3 Domain Waktu dan Domain Frekuensi

Sebuah sinyal,

$$x(t) = A \cos (2 \pi f t + \theta)$$

dapat digambarkan dengan dua cara yaitu dalam ***domain waktu*** dan dalam ***domain frekuensi***.

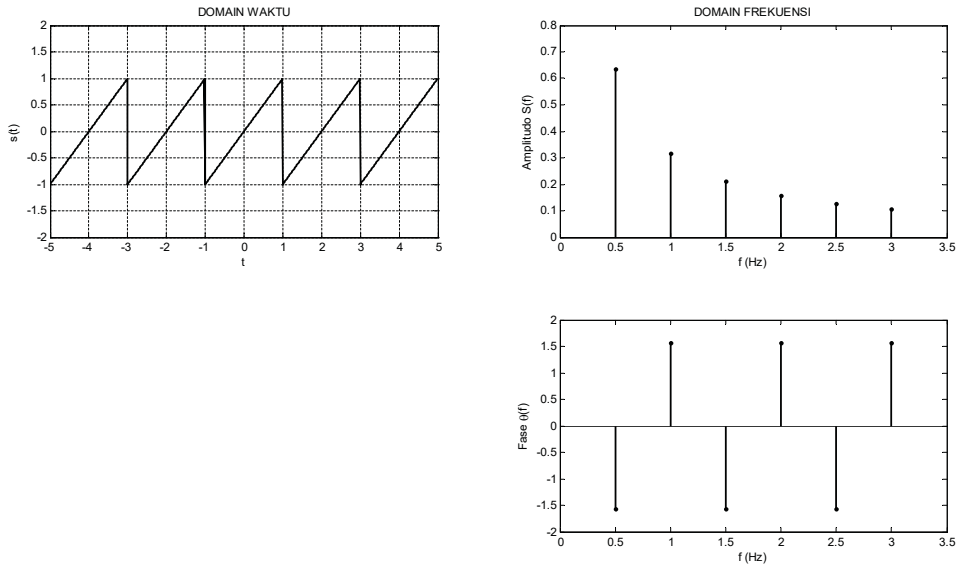
Sebagai contoh, sinyal $x(t) = 3 \cos (2 \pi 10 t - 0.2\pi)$ dapat digambarkan dalam domain waktu dan domain frekuensi seperti pada Gambar 2.7.



Gambar 2.7
Sinyal sinusoidal dalam domain waktu dan frekuensi

Gambar dalam domain frekuensi terdiri dari dua aksis yaitu axis amplitudo dan aksis fase. Pada aksis amplitudo, terdapat sebuah 'pentungan' pada 10 Hz dengan tinggi 3. Gambar ini menunjukkan bahwa sinyal tersebut adalah sinyal sinusiodal dengan amplitudo 3 dan frekuensi 10 Hz. Pada aksis fase terdapat satu 'pentungan' pada frekuensi 10 Hz setinggi -0.2π atau *-0.628*. Gambar ini menunjukkan bahwa komponen sinusoidal pada 10 Hz memiliki pergeseran fase -0.628 radian.

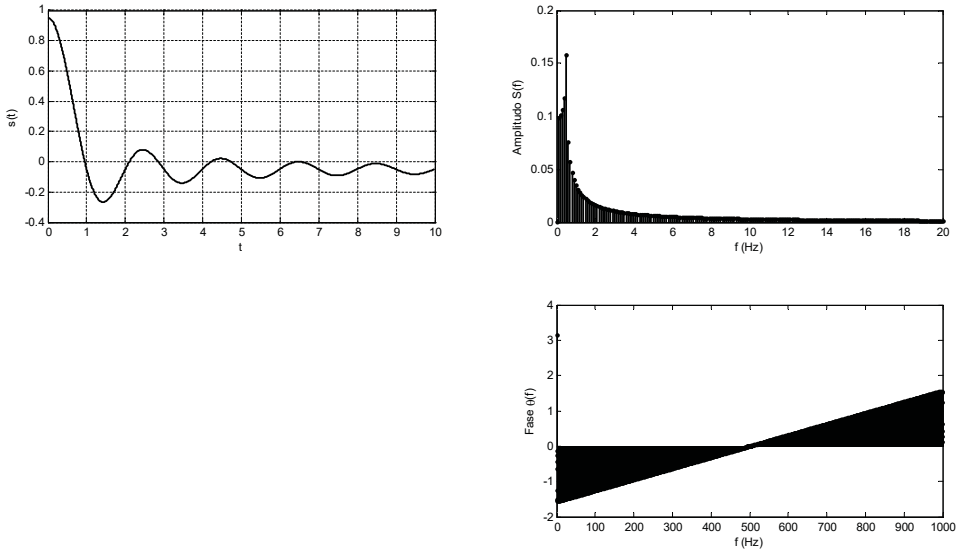
Gambar dalam domain frekuensi selalu mengasumsikan bahwa sinyal tersebut berbentuk sinusoidal (lebih tepatnya: ***kosinus***). Sehingga sinyal gigi gergaji, *s(t)*, pada Gambar 2.4 di atas akan digambarkan dalam domain frekuensi seperti pada Gambar 2.8.



Gambar 2.8
Sinyal gigi gergaji dalam domain waktu dan frekuensi

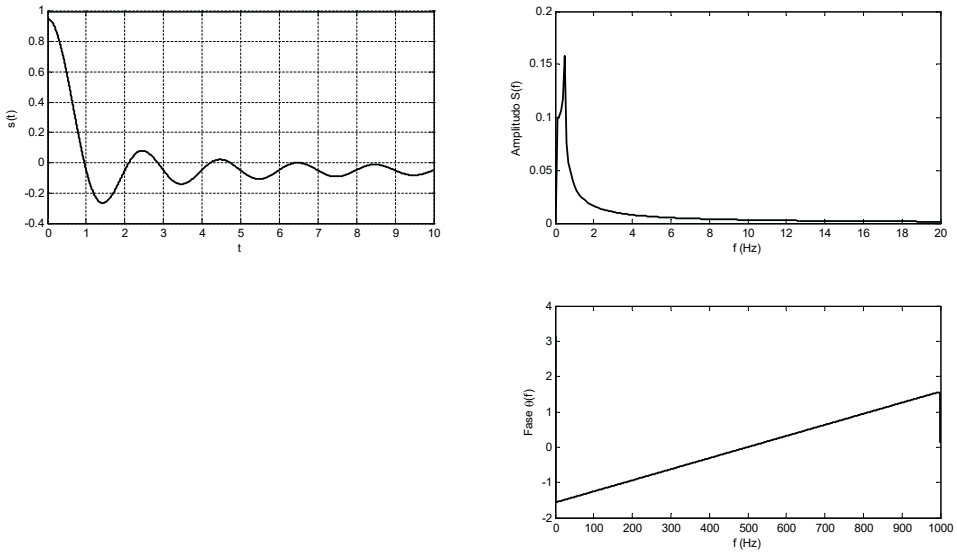
Gambar ini menunjukkan bahwa sinyal gigi gergaji memiliki paling tidak enam buah komponen sinusoidal dengan berbagai amplitudo dan fase. Jika kita hanya melihat pada gambar dalam domain frekuensi dan menutup gambar pada domain waktu maka kita akan mengalami kesulitan untuk memprediksi bahwa sinyal ini berbentuk gigi gergaji. Domain frekuensi selalu berasumsi bentuk sinyal sinusoidal saja.

Sinyal *s(t)* di atas hanya memiliki beberapa komponen sinusoidal saja. Jika kita memiliki suatu sinyal dengan bentuk yang kompleks (rumit) maka sinyal tersebut memiliki sangat banyak komponen sinusoidal seperti pada Gambar 2.9 berikut ini.



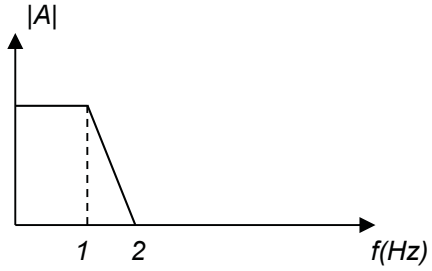
Gambar 2.9
Sinyal dengan banyak komponen sinusoidal

Gambar pada domain frekuensi memiliki sangat banyak garis yang cenderung terlihat sangat berdempetan. Untuk mempermudah penggambaran pada kasus ini maka domain frekuensi digambarkan dalam bentuk kurva seperti pada Gambar 2.10.



Gambar 2.10
Sinyal dengan banyak komponen sinusoidal digambarkan dengan kurva

Banyak kasus pemrosesan sinyal hanya memerlukan gambar amplitudo saja sehingga dalam buku ini kita lebih sering hanya akan menggambarkan kurva amplitudo dari domain frekuensi (spektrum frekuensi).



Gambar 2.11
Spektrum frekuensi

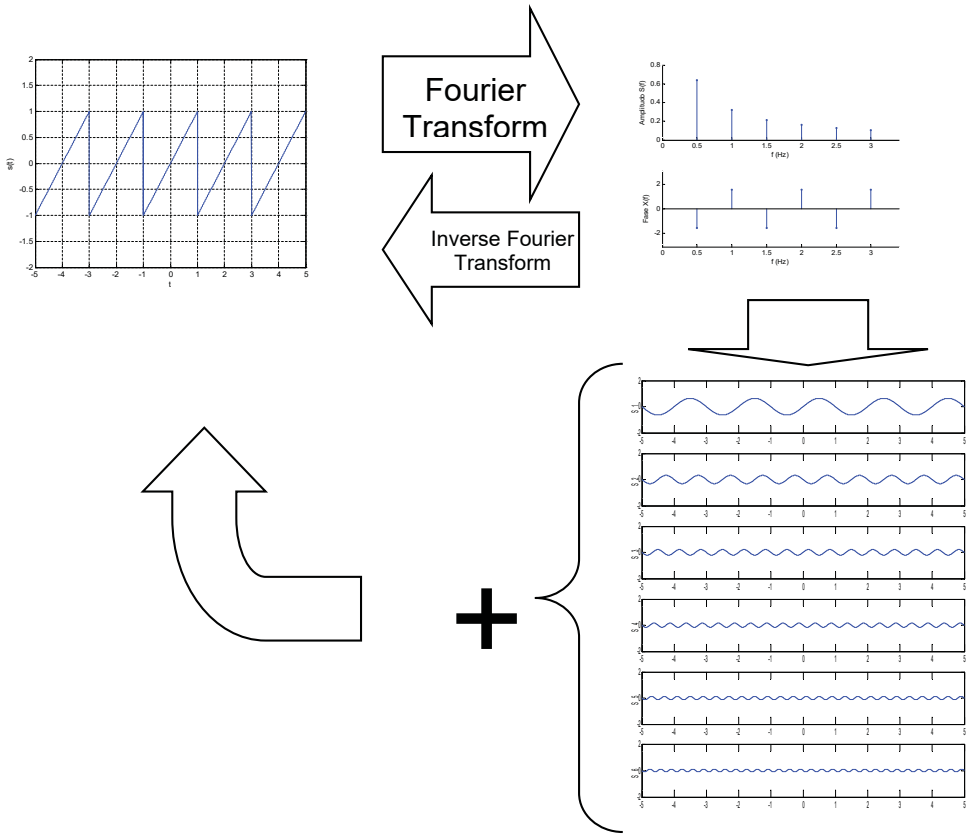
Dalam beberapa kasus, kita hanya memerlukan gambar spektrum frekuensi secara garis besar saja (*draft*) seperti pada Gambar 2.11.

Gambar ini sudah cukup untuk memberikan informasi bahwa sinyal ini memiliki banyak komponen sinusoidal pada daerah *0* sampai *1 Hz*, memiliki komponen sinusoidal pada frekuensi sampai dengan *2 Hz* tetapi dengan amplitudo yang rendah, dan tidak memiliki komponen sinusiodal dengan amplitudo yang signifikan pada frekuensi di atas *2 Hz*.

2.4 Transformasi Fourier

Transformasi Fourier (*Fourier Transform - FT*) adalah transformasi sinyal dari **domain waktu** ke **domain frekuensi**. Proses sebaliknya disebut *Inverse Fourier Transform (IFT)*.

Jika kita memiliki suatu sinyal dalam domain waktu, *s(t)*, maka kita dapat menggunakan Transformasi Fourier untuk menghitung/menggambar sinyal tersebut dalam domain frekuensi, *S(f)*. Gambar sinyal dalam domain frekuensi memberikan informasi kepada kita tentang komponen-komponen sinusoidal dari sinyal tersebut. Jika kita menjumlahkan komponen-komponen sinusoidal tersebut maka kita akan mendapatkan kembali sinyal *s(t)* lagi. Proses ini ditunjukkan pada Gambar 2.11.

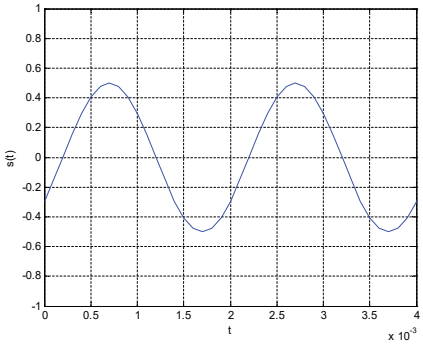


Gambar 2.11
Transformasi Fourier dan komponen sinusoidal

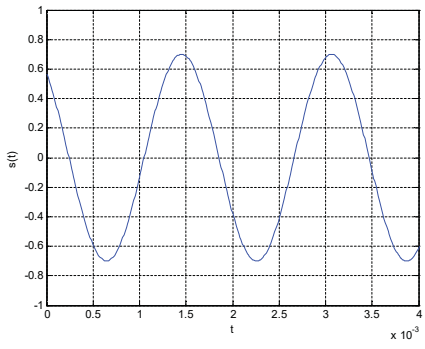
SOAL LATIHAN

1. Gambarlah sinyal-sinyal berikut pada kertas grafik:
   a. $x(t) = 0.5 \sin (2 \pi 25 t + 0.785)$
   b. $v(t) = 4.1 \cos (2 \pi 200 t - 0.25\pi)$
   c. $s(t) = 0.3 \cos (500 t - 0.5)$
   d. $i(t) = -0.5 \sin (400 t + 30^{o})$

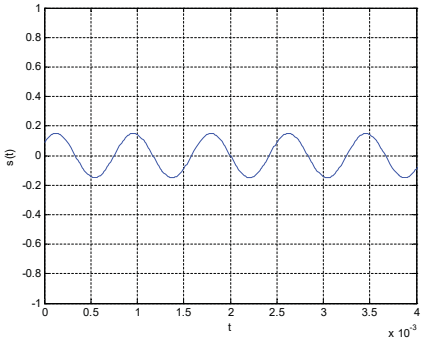
2. Tulislah persamaan dari sinyal-sinyal berikut ini dalam bentuk sinus dan kosinus:



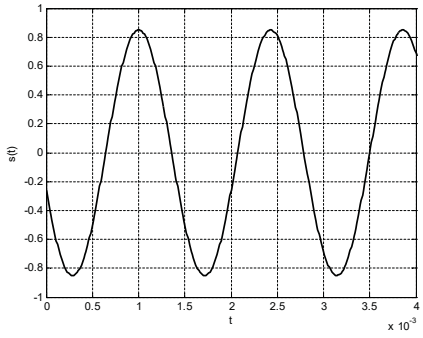
a



b



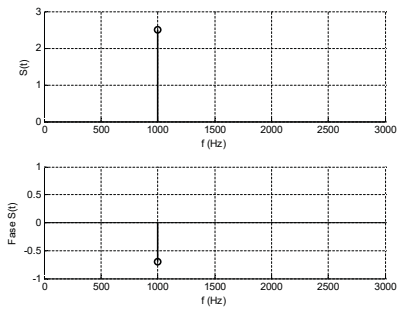
c



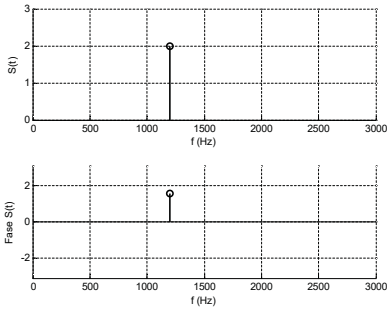
d

3. Gambarlah sinyal-sinyal pada soal nomor 2 dalam domain frekuensi.

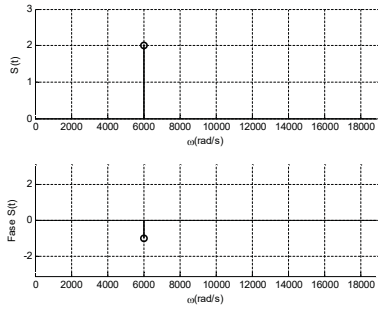
4. Tulislah persamaan sinyal berikut ini dan gambarlah dalam domain waktu



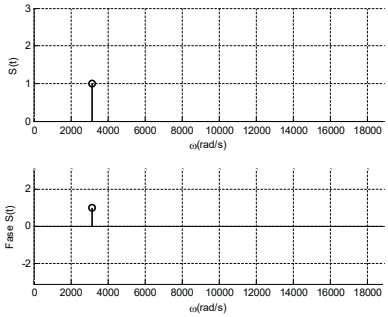
a



b



c



d

5. Gunakan program `modul2_5.m` untuk melihat bentuk dari sinyal yang tersimpan dalam file `sinyal2_5.mat` dan melihat sinyal tersebut dalam domain frekuensi. Buatlah tiruan dari sinyal tersebut dengan menjumlahkan satu atau banyak sinyal sinusoidal yang merupakan komponen dari sinyal tersebut. Bandingkan sinyal tiruan anda dengan sinyal aslinya.

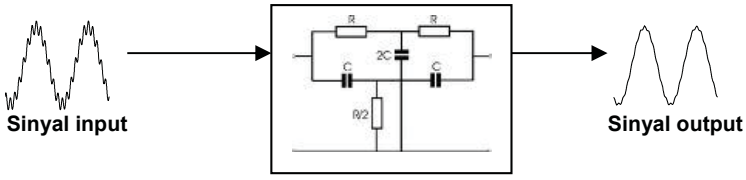
6. Bukalah program `modul2_5.m` dan gantilah nama sinyal untuk melihat bentuk dari sinyal yang tersimpan dalam file `sinyal2_6.mat`, dan melihat sinyal tersebut dalam domain frekuensi. Buatlah tiruan dari sinyal tersebut dengan menjumlahkan satu

atau banyak sinyal sinusoidal yang merupakan komponen utama dari sinyal tersebut. Bandingkan sinyal tiruan anda dengan sinyal aslinya.

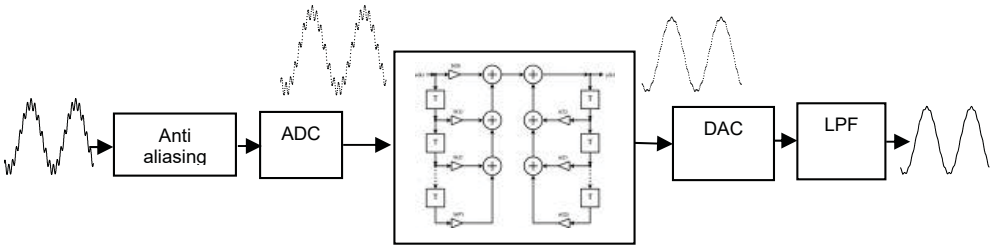
7. Sinyal yang direkam dalam file `sinyal2_7.mat` merupakan suara dari klakson kapal Dengarkanlah bunyinya dengan menggunakan perintah 'sound'. Gunakan Fourier Transform (dengan menggunakan program `modul2_5.m`) untuk mengetahui komponen utama dari sinyal tersebut, kemudian buatlah sinyal tiruanya dengan menjumlahkan berbagai sinyal sinusoidal. Bandingkan bunyi dari sinyal tiruan ini dengan sinyal aslinya. Usahakan agar sinyal tiruan yang anda buat menyerupai bunyi aslinya.

3
Sinyal Diskrit

Gambar 3.1 menunjukkan diagram blok dari sistem pemrosesan sinyal analog, sedangkan Gambar 3.2 menunjukkan diagram blok dari sistem pemrosesan sinyal digital (Digital Signal Processing – DSP).



Gambar 3.1
Sistem Pemrosesan Sinyal Analog



Gambar 3.2
Sistem Pemrosesan Sinyal Digital

Semua sinyal nyata dalam dunia kita bersifat analog. Sistem pemrosesan sinyal analog dapat langsung memproses sinyal analog karena sistem pemrosesan sinyal analog menggunakan komponen analog. Sistem pemrosesan sinyal analog berisi komponen-komponen elektronika seperti resistor, kapasitor, induktor, transistor, dan penguat operasional.

Sistem pemrosesan sinyal digital memproses sinyal secara numerik (proses aritmetika) sehingga sinyal input yang bersifat analog harus terlebih dahulu diubah menjadi sinyal diskrit (angka-angka) melalui proses ADC (Analog to Digital Converter). Setelah diproses secara digital, data diskrit yang dihasilkan pada bagian output harus diubah kembali menjadi sinyal analog dengan menggunakan DAC (Digital to Analog Converter). Proses DAC/ADC memerlukan filter anti aliasing dan filter rekonstruksi (berupa LPF). Kedua filter ini akan dijelaskan kemudian.

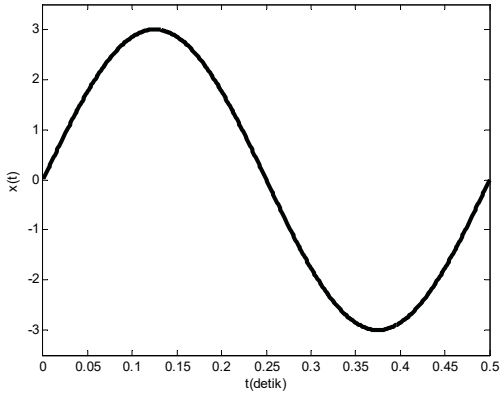
Penjelasan pada bab ini bertujuan untuk memberikan pemahaman tentang konsep sinyal diskrit, penulisan dan ekspresi matematisnya, serta proses diskritisasi sinyal. Penjelasan ini juga memberikan pemahaman tentang aliasing dan batasan dari proses diskritisasi suatu sinyal.

Setelah mempelajari materi dalam bab ini maka mahasiswa akan dapat mengidentifikasi parameter dari sinyal diskrit, menuliskan persamaan sinyal diskrit, dan menggambar sinyal diskrit.

3.1 Definisi Sinyal Diskrit

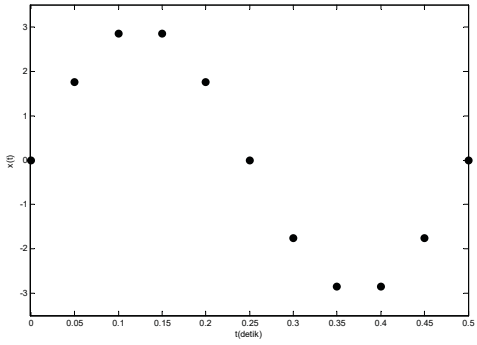
Sebuah sinyal analog, *s(t)*, memiliki nilai di semua *t*. Sinyal diskrit hanya memiliki nilai di nilai *t* tertentu saja. Sinyal diskrit dapat dilihat sebagai cuplikan titik-titik pada suatu sinyal analog.

Gambar 3.3 adalah sinyal analog $x(t) = 3 \sin (2\pi\, 2\, t)$ dengan frekuensi, $f = 2$ *Hz*, dan amplotudo, $A = 3$.



Gambar 3.3
Sinyal Analog

Sedangkan Gambar 3.4 adalah sampling dari sinyal *x(t)* tersebut dengan jarak antara sample (periode sampling) sebesar, $T_s = 0.05$ detik.



Gambar 3.4
Sampling dari Sinyal *x(t)* di Gambar 3.3

Proses sampling ini menghasilkan sinyal diskrit *x(n)* untuk $0 \leq n \leq 10$ sebagai berikut:

$x(0) = 0$
$x(1) = 1.7635$
$x(2) = 2.8532$
$x(3) = 2.8532$
$x(4) = 1.7635$
$x(5) = 0$
$x(6) = -1.7635$
$x(7) = -2.8532$
$x(8) = -2.8532$
$x(9) = -1.7635$
$x(10) = 0$

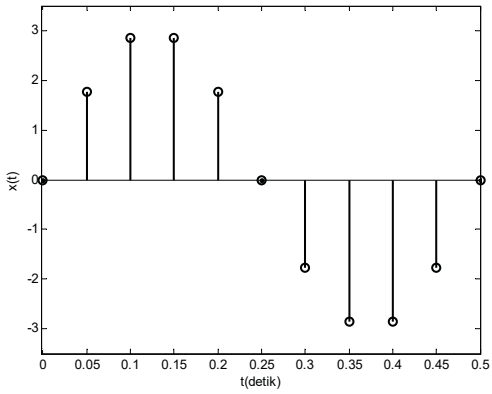
Angka-angka ini dapat dihitung dengan ekspresi diskrit:

$$x(n) = 3 \sin (2 \pi 10 n T_s)$$

atau

$$x(n) = 3 \sin (2 \pi 0.5 n)$$

Pada umumnya sinyal diskrit digambar berupa *'stem'* agar lebih mudah terlihat. Gambar 3.5 menunjukan sinyal diskrit dari Gambar 3.3.



Gambar 3.5
Sinyal diskrit *x(n)*

Jika kita menyimpan sinyal diskrit tersebut dalam komputer maka yang perlu disimpan adalah angka-angka dari *x(0)* sampai *x(10)* dan frekuensi sampling f_s. Frekuensi sampling adalah banyaknya titik sampling dalam satu detik. Pada sinyal ini

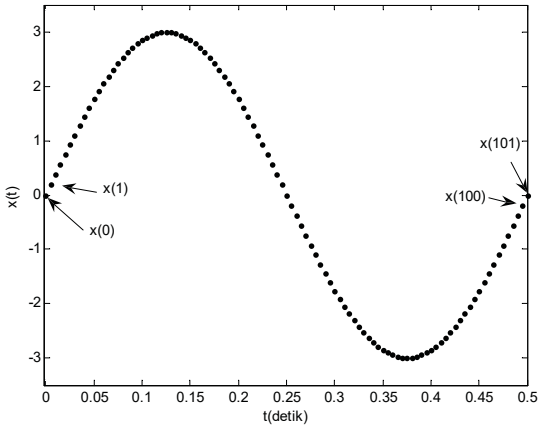
$$f_s = \frac{1}{T_s} = \frac{1}{0.05\,\text{det}} = 20\,Hz$$

Secara umum, data yang disimpan adalah *x(n)* untuk $0 \leq n \leq N$. Namun kebanyakan software simulasi (termasuk Matlab) tidak mengenal indeks ke-0, jadi tidak mengenal *x(0)*, sehingga untuk data *x(n)* di atas kita menggunakan *x(n)* untuk $1 \leq n \leq 11$.

Ketika disimpan di dalam komputer, angka-angka dari sinyal diskrit ini kemudian harus diubah menjadi angka digital oleh ADC untuk diolah secara digital dalam sistem DSP (komputer antau mikroprosesor).

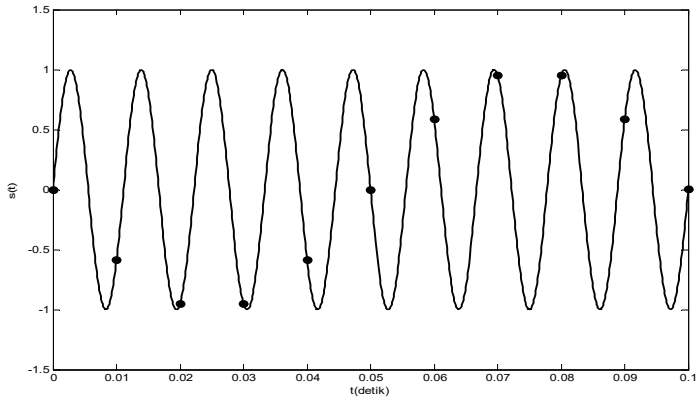
3.2 Frekuensi Sampling

Semakin besar frekuensi sampling maka titik-titik sampling tersebut akan semakin menggambarkan bentuk sesungguhnya dari sinyal analog, namun hal ini menyebabkan semakin banyak data yang harus disimpan dalam sistem digital. Jika sinyal di Gambar 3.3 disampling dengan $f_s = 20\ Hz$ maka kita hanya perlu menyimpan *11* data, tetapi jika disampling dengan $f_s = 200\ Hz$ maka titik-titik samplingnya (Gambar 3.6) lebih menyerupai sinyal aslinya tetapi kita harus menyimpan 101 data.



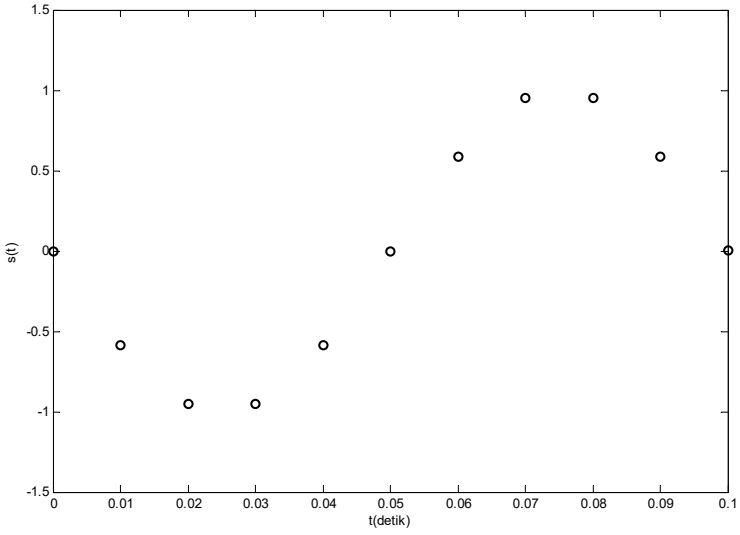
Gambar 3.6
Sampling dari Sinyal *x(t)* dengan $f_s = 200$ Hz

Frekuensi sampling paling kecil adalah 2 kali frekuensi dari sinyal yang disampling. Jika frekuensi sampling lebih kecil dari 2 kali frekuensi sinyal maka titik-titik sampling tersebut akan merepresentasikan sinyal yang lain. Gambar 3.7 menunjukkan sebuah sinyal *s(t)* dengan frekuensi *90 Hz*, dengan titik-titik sampling yang disampling menggunakan *fs = 100 Hz*.



Gambar 3.7
Sinyal *s(t)* dengan *f = 90 Hz* yang disampling dengan $f_s = 100$ Hz

Titik-titik sampling tersebut ternyata merepresentasikan sinyal lain seperti ditunjukkan pada Gambar 3.8. Artinya, jika frekuensi sampling yang digunakan kurang dari 2 kali frekuensi sinyal maka titik-titik sampling tersebut tidak akan dengan tepat merepresentasikan sinyal aslinya.

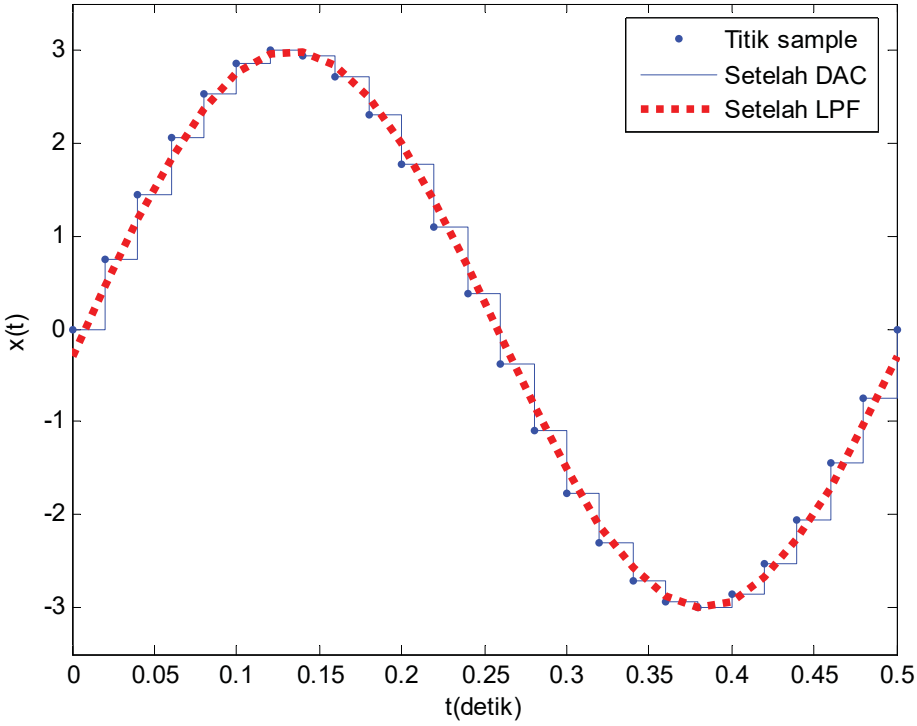


Gambar 3.8
Titik-titik sampling sinyal *s(t)* dapat salah diinterpretasi menjadi sinyal lain

Gambar 3.8 tersebut seolah-olah merupakan sampling dari sinyal lain dengan frekuensi 10 Hz. Peristiwa ini disebut aliasing, yaitu jika sinyal yang disampling memiliki frekuensi lebih besar dari 0.5 f_s. Untuk menghindari kejadian ini maka setiap sistem digital menyertakan filter anti aliasing filter sebelum proses ADC. Filter ini pasti berupa filter analog karena sinyalnya belum disampling. Filter ini berfungsi untuk menghentikan semua komponen sinusoidal dari sinyal input dengan frekuensi lebih besar dari $0.5\ f_s$.

3.3 Proses DAC

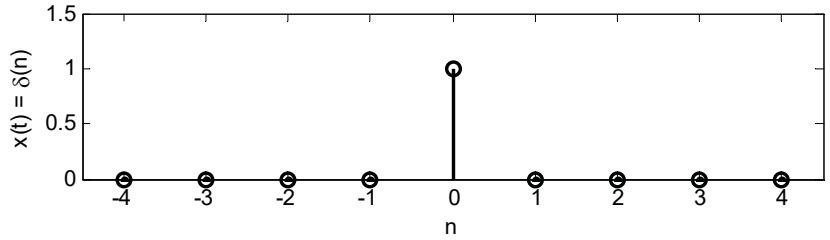
Proses DAC adalah proses untuk mengubah kembali sinyal digital menjadi sinyal analog. Pada proses DAC titik-titik sampling tersebut akan diubah menjadi sinyal analog dalam bentuk sinyal tangga. Sinyal ini kemudian difilter dengan Low Pass Filter untuk menghilangkan bentuk tangga. Gambar 3.9 menunjukkan proses DAC dari sampling pada Gambar 3.3 yang disampling dengan $f_s = 50\ Hz$.



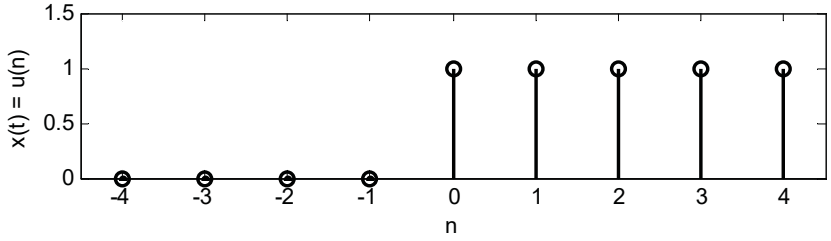
Gambar 3.9
Proses DAC

3.4 Sinyal-sinyal diskrit

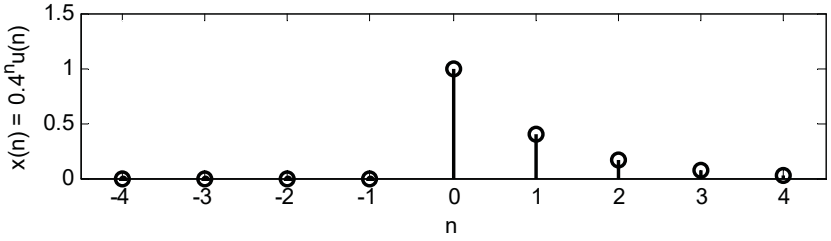
Beberapa sinyal diskrit yang umum ditemukan selain sinyal sinusoidal adalah sinyal impulse (Gambar 3.10), sinyal tangga atau step (Gambar 3.11), dan sinyal eksponensial (Gambar 3.12). Sinyal impulse adalah sinyal yang bernilai *1* (satu) pada saat $t = 0$ dan bernilai *0* (nol) pada semua titik sample yang lain. Sinyal step adalah sinyal yang bernilai *1* (satu) untuk $t \geq 0$. Sinyal eksponensial bernilai *1* (satu) pada $t = 0$ dan menurun secara eksponensial untuk $t > 0$.



Gambar 3.10
Sinyal impulse x(n) = δ(n)



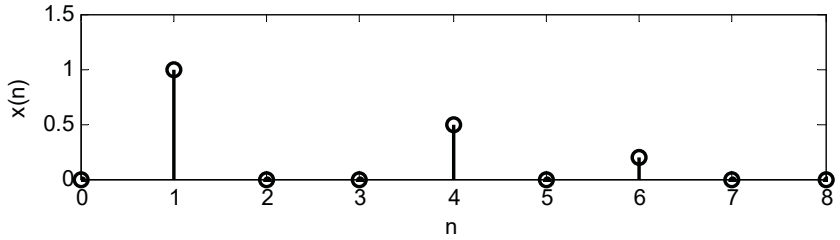
Gambar 3.11
Sinyal step x(n) = u(n)



Gambar 3.12
Sinyal $x(n) = a^n u(n)$ dengan a = 0.4.

Sinyal diskrit dapat ditulis dalam bentuk gambungan dari banyak sinyal impulse. Sinyal pada Gambar 3.13 dapat ditulis:

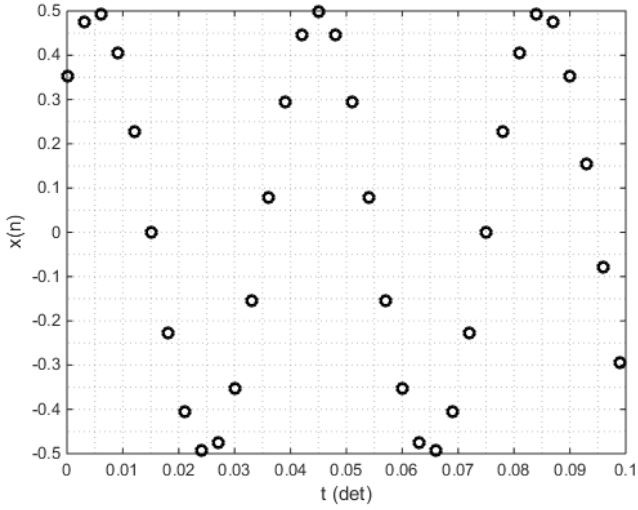
$$x(n) = \delta(n-1) + 0.5\,\delta(n-4) + 0.2\,\delta(n-6)$$



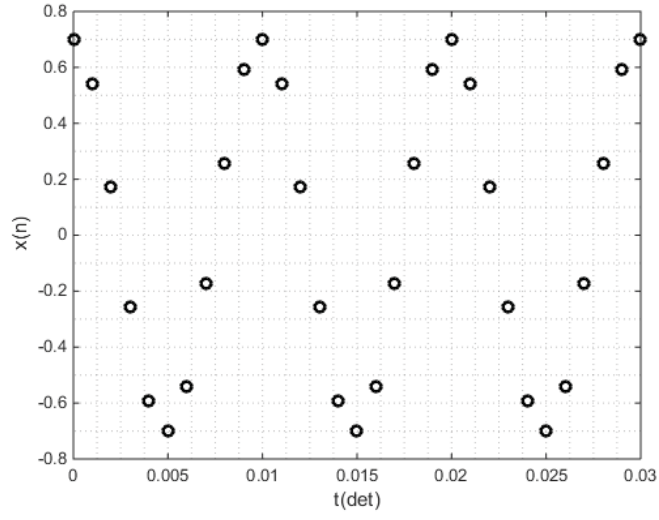
Gambar 3.13
Contoh sinyal diskrit x(n)

SOAL LATIHAN

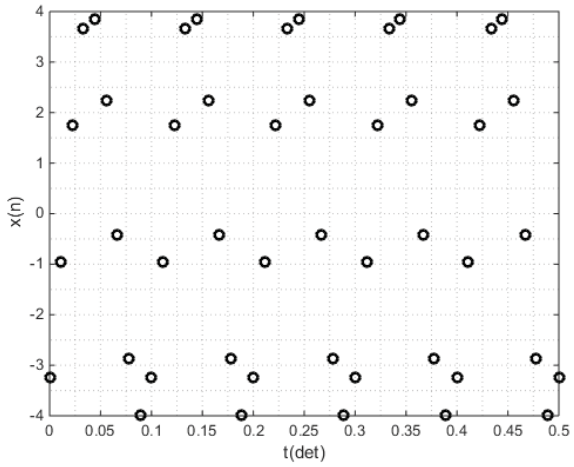
1. Amati gambar-gambar berikut ini dan tulislah persamaan *x(t)* dan *x(n)*, nilai dari *x(n)* untuk 20 data pertama, tentukan amplitudo, frekuensi dari sinyal (*f*), dan frekuensi sampling yang digunakan (f_s).



a



b



c

2. Untuk data *y(n)* sebagai berikut, gambarlah sample dari sinyal tersebut dengan $T_s = 3$ *mdet*, gambarlah *y(t)*, tulislah persamaan *y(n)* dan *y(t)*.

n	*y(n)*
0	0.68605
1	0.58668
2	0.40491
3	0.16627
4	-0.095717
5	-0.34426
6	-0.54446
7	-0.66818
8	-0.69807
9	-0.62991
10	-0.47328
11	-0.25018
12	0.0080529
13	0.26516
14	0.48502
15	0.63677
16	0.69908
17	0.66321
18	0.53419
19	0.33015
20	0.079738
21	-0.18187
22	-0.41794
23	-0.59531
24	-0.68906
25	-0.68605

3. Untuk data *s(n)* sebagai berikut, gambarlah sample dari sinyal tersebut dengan $f_s = 25\ Hz$, gambarlah *s(t)*.

n	*s(n)*
0	0.9845
1	0.5979
2	-0.0089
3	-0.3196
4	-0.2972
5	-0.3383
6	-0.6343
7	-0.8458
8	-0.5540
9	0.1288
10	0.6428
11	0.6591
12	0.4400

4. Untuk $x(n) = 0.5 \sin (1.6\ \pi\ n)$ dengan $f_s = 25\ Hz$, gambarlah titik-titik sample dan gambarlah juga *x(t)* dari sinyal tersebut. Tulislah persamaan *x(t)*.

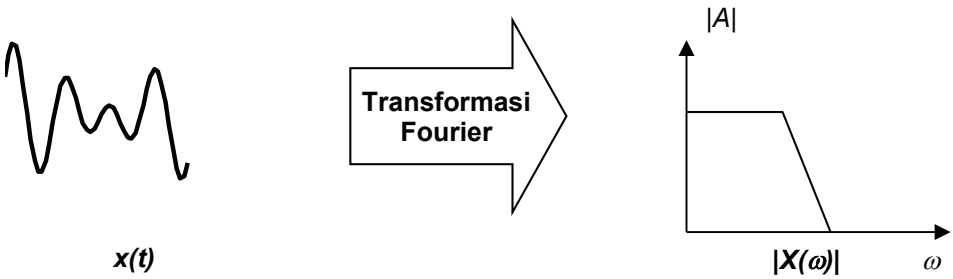
5. Hitunglah nilai dari sinyal diskrit berikut ini untuk $N = 10$ dan gambarlah sinyal-sinyal tersebut:
   a. $x(n) = 0.5 \sin (1.6\ \pi\ n)$ $\quad 0 \leq n \leq N$
   b. $y(n) = \cos (\pi\ n/2) \cos (\pi\ n/4)$ $\quad 0 \leq n \leq N$
   c. $s(n) = 0.7^{\,n}\ u(n-3)$ $\quad 0 \leq n \leq N$
   d. $s(n) = [u(n-3)-u(n-6)]$ $\quad 0 \leq n \leq N$
   e. $s(n) = 0.3^{\,n}\ [u(n)-u(n-5)]$ $\quad 0 \leq n \leq N$
   f. $s(n) = \delta(n-4)$ $\quad 0 \leq n \leq N$
   g. $s(n) = \delta(n-3) - \delta(n-7)$ $\quad 0 \leq n \leq N$

6. Gambarlah sinyal diskrit berikut dengan menggunakan Matlab:
   a. $x(n) = 0.5 \sin (1.6\ \pi\ n)$ $\quad 0 \leq n \leq 50$
   b. $y(n) = \cos (\pi\ n/2) \cos (\pi\ n/4)$ $\quad 0 \leq n \leq 100$

7. Gambarlah sample dari sinyal berikut dengan menggunakan Matlab (amati apakah terjadi aliasing):
   a. $x(t) = 2.5 \sin (2\ \pi\ 10\ t)$ $\quad f_s = 100\ Hz$
   b. $x(t) = 2.5 \sin (2\ \pi\ 10\ t)$ $\quad f_s = 50\ Hz$
   c. $x(t) = 2.5 \sin (2\ \pi\ 10\ t)$ $\quad f_s = 25\ Hz$
   d. $x(t) = 2.5 \sin (2\ \pi\ 10\ t)$ $\quad f_s = 15\ Hz$

4
Discrete Fourier Transform

Bab ini membahas konsep Discrete Fourier Transform (DFT), metode menghitung DFT, dan Fast Fourier Transform (FFT). Bab ini bertujuan untuk memberikan pemahaman tentang DFT sehingga mahasiswa akan dapat melakukan perhitungan DFT, menghitung magnitudo, frekuensi dan fase dari hasil perhitungan DFT, serta melakukan perhitungan dengan metode FFT. Bab ini juga melangkapi mahasiswa dengan kemampuan untuk menghitung spektrum frekuensi.

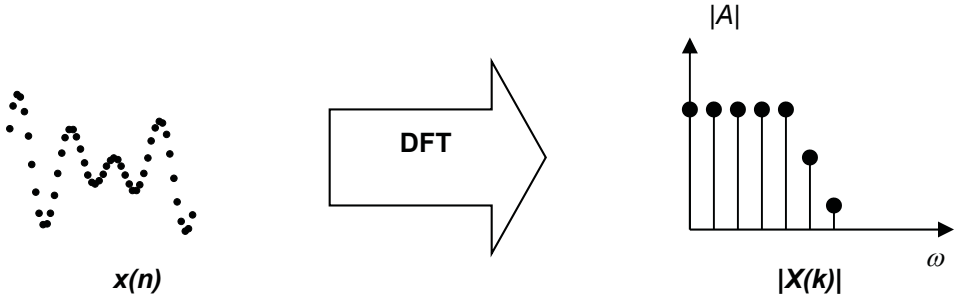


Gambar 4.1
Transformasi Fourier

Bab 2 telah menjelaskan tentang sinyal dalam domain waktu dan domain frekuensi. Transformasi Fourier digunakan untuk

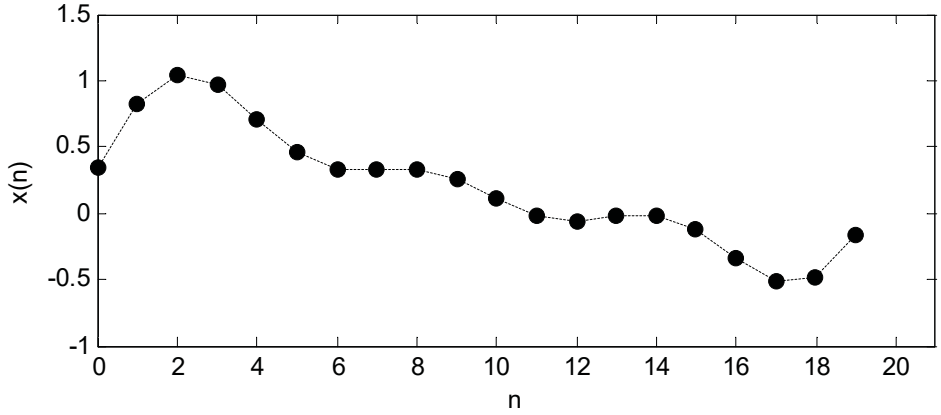
mentransformasi sinyal analog dari domain waktu ke domain frekuensi (spektrum frekuensi) seperti ditunjukkan pada Gambar 4.1.

Discrete Fourier Transform (DFT) adalah bentuk diskrit dari transformasi Fourier. DFT mentransformasi sinyal diskrit dari domain waktu ke domain frekuensi (juga dalam bentuk diskrit). Hal ini ditunjukkan pada Gambar 4.2.

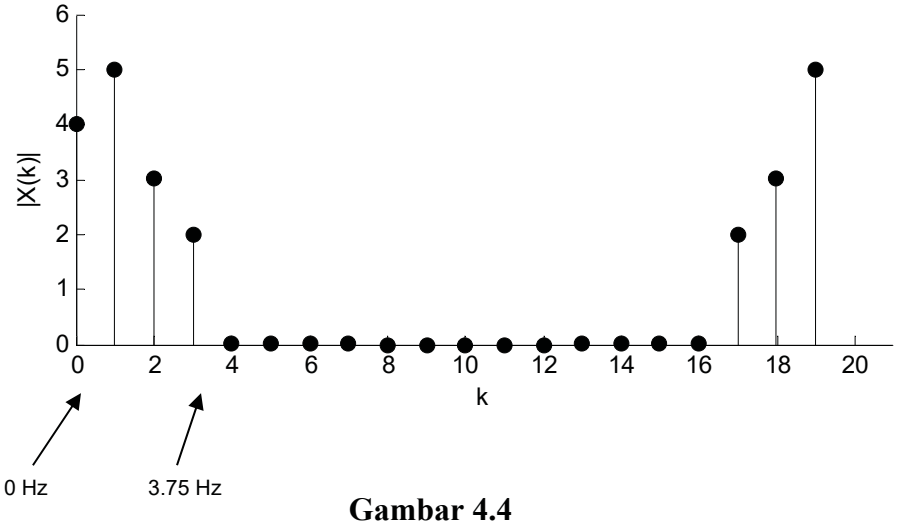


Gambar 4.2
Discrete Fourier Transform

Ketika sebuah sinyal diskrit, seperti pada Gambar 4.3, ditransformasi dengan menggunakan DFT, maka akan dihasilkan spektrum frekuensi dalam bentuk diskrit seperti pada Gambar 4.4.



Gambar 4.3
Sinyal Diskrit



Gambar 4.4
Hasil perhitungan DFT dari sinyal pada Gambar 4.3 ($f_s = 25$ *Hz*)

Beberapa hal penting yang dapat diamati dari hasil perhitungan DFT di Gambar 4.4 adalah:

- Sinyal dalam domain waktu ditulis dengan huruf kecil (mis. *x(n)*), sedangkan hasil perhitungan DFT berada dalam domain frekuensi sehingga ditulis dengan huruf besar (mis. *X(k)*).
- Nomor sample pada sinyal input (domain waktu) ditulis dengan menggunakan huruf *n* sedangkan nomor sample di domain frekuensi menggunakan huruf *k*.
- Sinyal input, *x(n)*, memiliki 20 sample, maka hasil perhitungan DFT, *X(k)*, juga memiliki 20 buah sample. Jika sinyal *x(n)* memiliki *N+1* buah sample maka perhitungan DFT menghasilkan *K+1* sample, dimana $K = N$.
- Setiap nilai *k* mewakili nilai frekuensi tertentu tergantung pada frekuensi sampling, f_s, dari sinyal *x(n)*. Jika $f_s = 25$ *Hz* maka.

k = 0 $f = 0\ Hz$

k = 1 $f = \frac{f_s}{jumlah\ \ sample} k = \frac{25\ Hz}{20}(1) = 1.25\ \ Hz$

k = 2 $f = \frac{25\ Hz}{20}(2) = 2.50\ \ Hz$

k = 3 $f = \frac{25\ Hz}{20}(3) = 3.75\ \ Hz$

...

k = 19 $f = \frac{25\ Hz}{20}(19) = 23.75\ \ Hz$

- $|X(k)|$ memiliki bentuk yang simetris (pencerminan) pada $k=10$ (setara $f_s/2$). Jadi

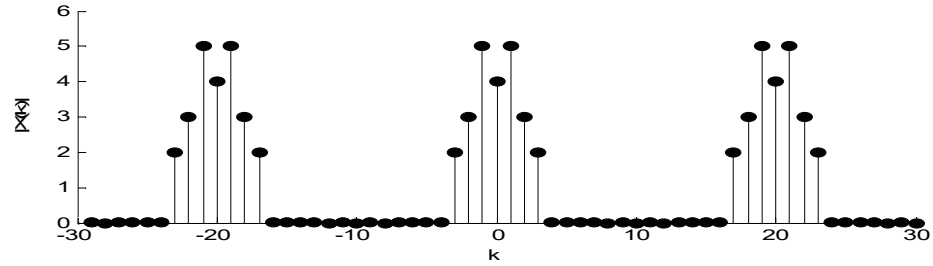
$$|X(19)| = |X(1)|$$
$$|X(18)| = |X(2)|$$
$$|X(17)| = |X(3)|$$
$$|X(16)| = |X(4)|$$
$$|X(15)| = |X(5)|$$
$$|X(14)| = |X(6)|$$
$$\ldots$$
$$|X(11)| = |X(9)|$$

Maka jika kita menganalisa hasil dari DFT sinyal tersebut, kita cukup menggunakan $|X(k)|$ untuk $0 \leq k \leq 10$ atau dengan kata lain dari *0 Hz* sampai *12.5 Hz* yaitu $f_s / 2$.

Jika kita menghitung DFT dari sinyal $x(n)$ untuk $0 \leq n \leq N$, maka hasil DFT yang digunakan adalah $X(k)$ untuk $0 \leq k \leq K/2$, dimana $K = N$.

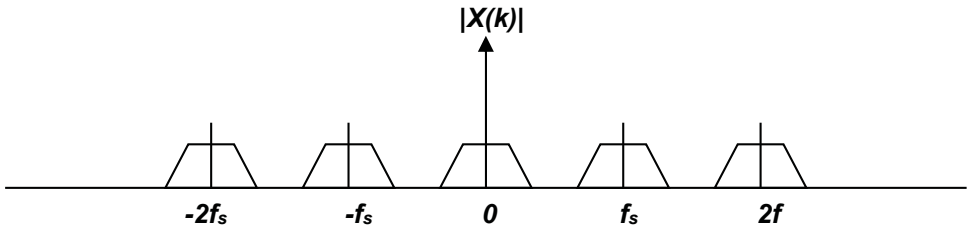
Hal ini dapat juga menjelaskan mengapa f_s harus lebih besar dari dua kali frekuensi sinyal. Jika sinyal di atas memiliki frekuensi lebih besar dari $f_s/2$ (*12.5 Hz*) maka akan digambar pada $k > 10$ padahal nilai $|X(k)|$ untuk $k > 10$ seharusnya hanya merupakan pencerminan dari $|X(k)|$ yang lebih kecil dari *10*.

Sesungguhnya pencerminan ini terjadi secara berulang. Jika kita dapat menghitung $|X(k)|$ untuk nilai k yang lain maka akan menghasilkan pengulangan seperti pada Gambar 4.5.



Gambar 4.5
Hasil dari DFT sinyal di Gambar 4.3 untuk $-30 \leq k \leq 30$

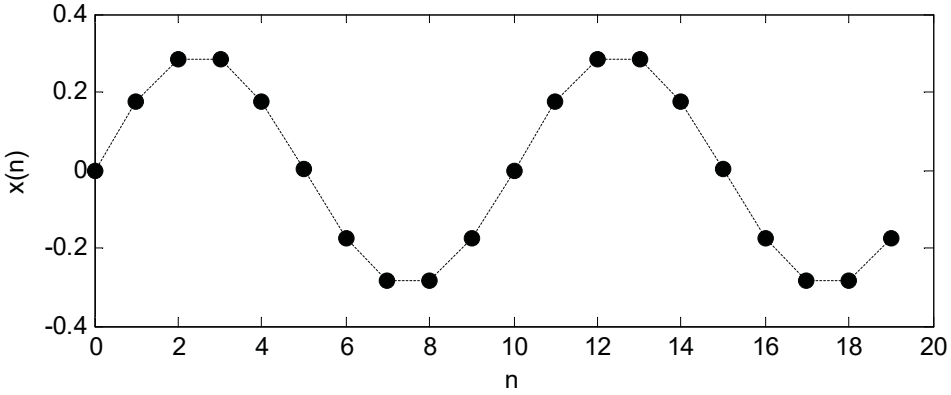
Sehingga hasil dari DFT sering digambarkan secara *draft* seperti pada gambar berikut ini.



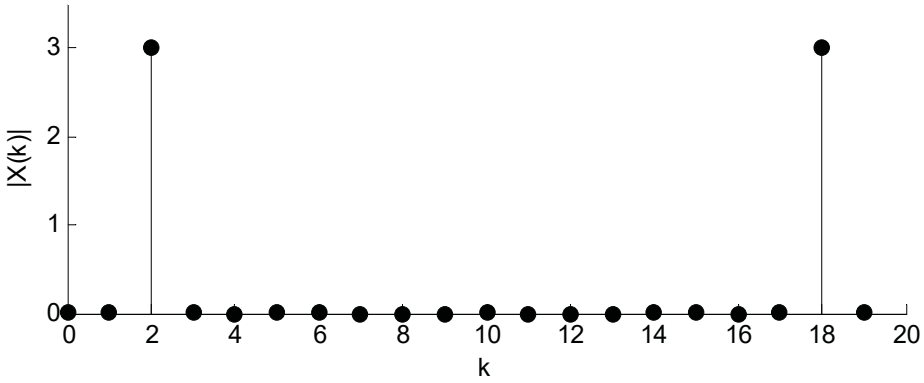
Gambar 4.6
***Draft* spektrum diskrit**

4.1 Informasi amplitudo dalam DFT

Jika kita memiliki suatu sinyal sinusoidal dengan amplitudo *0.3*, frekuensi *2.5 Hz* dan frekuensi sampling *25 Hz* seperti digambarkan pada Gambar 4.7, maka hasil perhitungan DFT dari sinyal tersebut akan terlihat seperti pada Gambar 4.8.



Gambar 4.7
Sinyal diskrit 2.5 Hz dengan fs = 25 Hz.

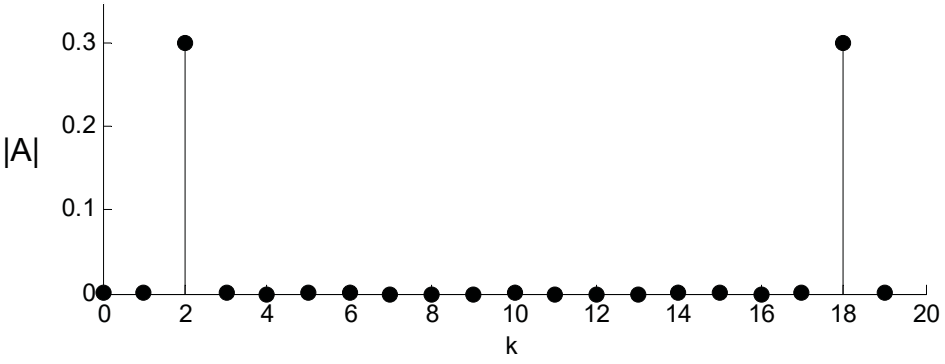


Gambar 4.8
Hasil DFT dari sinyal pada Gambar 4.7.

Dari spektrum |*X(k)*| tersebut kita dapat mengetahui bahwa frekuensi sinyal yang ditransformasi adalah:

$$f = \frac{f_s}{jumlah\ sample} k = \frac{25\,Hz}{20}(2) = 2.5\ Hz$$

Data ini tepat sama dengan komponen sinusiodal dari sinyal input yang diberikan. Amplitudo dari komponen sinusoidal tersebut dapat kita peroleh dari tinggi *'pentungan'* pada frekuensi tersebut, *k = 2*. Tetapi informasi mengenai amplitudo dari sinyal tersebut tidak tepat. Gambar 4.8 menunjukkan bahwa amplotudo sinyal sebesar 3, padahal sesungguhnya hanya 0.3. Untuk mengetahui amplitudo sesungguhnya, hasil dari DFT harus dinormalisasi dengan pengali $\frac{2}{jumlah\ sample}$ yaitu $\frac{2}{N+1}$ seperti pada Gambar 4.9 (kecuali *X(0)* yang hanya dikalikan dengan $\frac{1}{N+1}$). Hasil DFT yang sudah dinormalisasi ini disebut spektrum amplitudo (*Amplitude Spectrum*), |*A*|.



Gambar 4.9
Spektrum Amplitudo dari sinyal Gambar 4.7

4.2 Menghitung Spektrum frekuensi dengan DFT

Spektrum frekuensi dari suatu sinyal diskrit, *x(n)*, dapat dihitung dengan menggunakan rumus DFT:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\frac{2\pi k n}{N}}$$

Jika sinyal input, *x(n)*, ditransformasi dengan DFT maka akan menghasilkan spektrum dari sinyal itu, *X(k)*, yang masih dalam bentuk bilangan kompleks. Magnitudo dari spektrum frekuensi dihitung dengan:

$$\left|X(k)\right| = \sqrt{X(k)_{real}^{\ 2} + X(k)_{imajiner}^{\ 2}}$$

Sedangkan sudut fase dihitung dengan:

$$\varphi(k) = \tan^{-1} \frac{X(k)_{imajiner}}{X(k)_{real}}$$

Sebagai contoh, sinyal pada Gambar 4.3 memiliki nilai:

x(0)	=	0.3493
x(1)	=	0.8183
x(2)	=	1.0446
x(3)	=	0.9720
x(4)	=	0.7148
x(5)	=	0.4611
x(6)	=	0.3349
x(7)	=	0.3254
x(8)	=	0.3285
x(9)	=	0.2550
x(10)	=	0.1106
x(11)	=	-0.0189
x(12)	=	-0.0583
x(13)	=	-0.0227
x(14)	=	-0.0124
x(15)	=	-0.1210
x(16)	=	-0.3342

x(17) = -0.5117
x(18) = -0.4777
x(19) = -0.1574

Spektrum frekuensi dari sinyal tersebut yang dihitung dengan DFT adalah:

Hasil DFT	Amplitudo	Fase (rad)
X(0) = 4.0000	|X(0)| = 4.0000	φ(0) = 0
X(1) = 0.9933 + 4.9003 j	|X(1)| = 5.0000	φ(1) = -1.3708
X(2) = 0.2995 + 2.9850 j	|X(2)| = 3.0000	φ(2) = -1.4708
X(3) = 0.1997 + 1.9900 j	|X(3)| = 2.0000	φ(3) = -1.4708
X(4) = 0.0000 + 0.0000 j	|X(4)| = 0.0000	φ(4) = -1.0605
X(5) = 0.0000 + 0.0000 j	|X(5)| = 0.0000	φ(5) = -0.2663
X(6) = 0.0000 - 0.0000 j	|X(6)| = 0.0000	φ(6) = 0.3803
X(7) =-0.0000 - 0.0000 j	|X(7)| = 0.0000	φ(7) = 2.0344
X(8) = 0 - 0.0000 j	|X(8)| = 0.0000	φ(8) = 1.5708
X(9) =-0.0000 - 0.0000 j	|X(9)| = 0.0000	φ(9) = 1.9296
X(10) = 0	|X(10)| = 0	φ(10) = 0
X(11) =-0.0000 + 0.0000 j	|X(11)| = 0.0000	φ(11) = -1.9296
X(12) = 0 + 0.0000 j	|X(12)| = 0.0000	φ(12) = -1.5708
X(13) =-0.0000 + 0.0000 j	|X(13)| = 0.0000	φ(13) = -2.0344
X(14) = 0.0000 + 0.0000 j	|X(14)| = 0.0000	φ(14) = -0.3803
X(15) = 0.0000 - 0.0000 j	|X(15)| = 0.0000	φ(15) = 0.2663
X(16) = 0.0000 - 0.0000 j	|X(16)| = 0.0000	φ(16) = 1.0605
X(17) = 0.1997 - 1.9900 j	|X(17)| = 2.0000	φ(17) = 1.4708
X(18) = 0.2995 - 2.9850 j	|X(18)| = 3.0000	φ(18) = 1.4708
X(19) = 0.9933 - 4.9003 j	|X(19)| = 5.0000	φ(19) = 1.3708

Magnitudo dari spektrum frekuensi dihitung dengan:

$$\left|X(k)\right| = \sqrt{X(k)_{real}^{\ 2} + X(k)_{imajiner}^{\ 2}}$$

Sedangkan sudut fase dihitung dengan:

$$\varphi(k) = \tan^{-1} \frac{X(k)_{imajiner}}{X(k)_{real}}$$

Ada beberapa nilai fase yang terlihat aneh seperti *φ(4)* sampai *φ(16)*. Nilai-nilai ini tidak memiliki arti karena magnitudonya *0* (nol).

Menghitung spektrum frekuensi dengan persamaan DFT memerlukan sangat banyak proses perhitungan. Ada cara menghitung DFT yang lebih cepat. Cara ini disebut *Fast Fourier Transform* (FFT). Cara ini bisa digunakan bila jumlah sample sinyal input sebanyak 2, 4, 8, 16, atau 2^p (p = bilangan integer positif).

Perkembangan teknologi komputer telah membuat sehingga perhitungan DFT maupun FFT seperti ini dapat dilakukan dengan cepat. Pada bagian berikut ini kita akan belajar cara menghitung FFT untuk 4 dan 8 sample. Tentunya kita harus menggunakan komputer untuk jumlah sample yang lebih banyak.

4.3 Fast Fourier Transform (FFT)

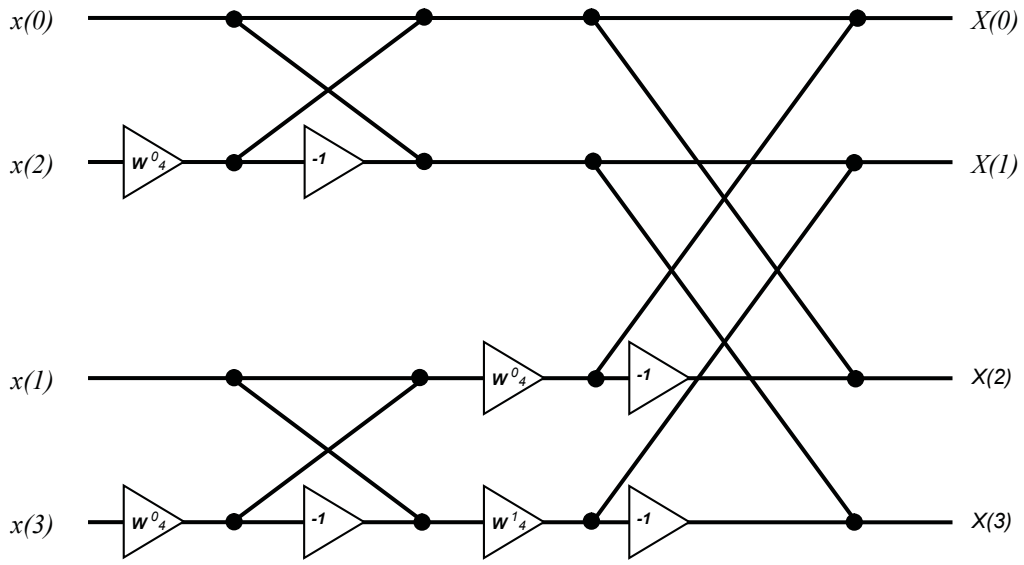
FFT dari suatu sinyal, *x(n)*, dihitung dengan:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

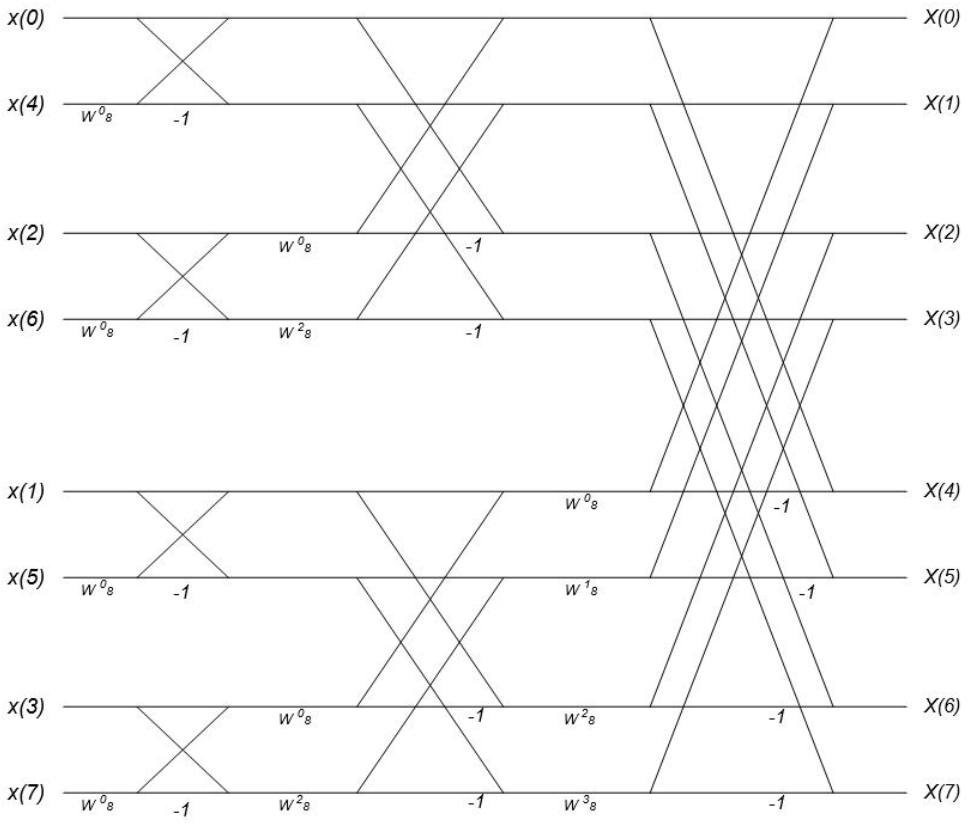
dimana

$$W_N^{nk} = e^{-j\frac{2\pi k n}{N}}$$

FFT ini dihitung dengan metode *Butterfly*. Metode ini memiliki beberapa langkah yaitu menggambar diagram *butterfly*, menempatkan input, dan menghitung pada setiap tahap. Gambar dari diagram *butterfly* untuk FFT dengan 4 sample adalah seperti pada Gambar 4.10, sedangkan untuk 8 input ditunjukkan pada Gambar 4.11.



Gambar 4.10
Diagram *butterfly* untuk 4 input



Gambar 4.11
Diagram *butterfly* untuk 8 input

Bagian output dari diagram ini disusun berurut *X(0)*, *X(1)*, ..., *X(7)* dari atas ke bawah, sedangkan bagian input tidak disusun berurut. Indeks bagian input disusun dengan metode *reverse bit* seperti pada tabel berikut:

Tabel 4.1
Indeks Inverse Bit

Indeks	Binary	Inverse Bit (urutan binary dibalik)	Indeks inverse bit
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

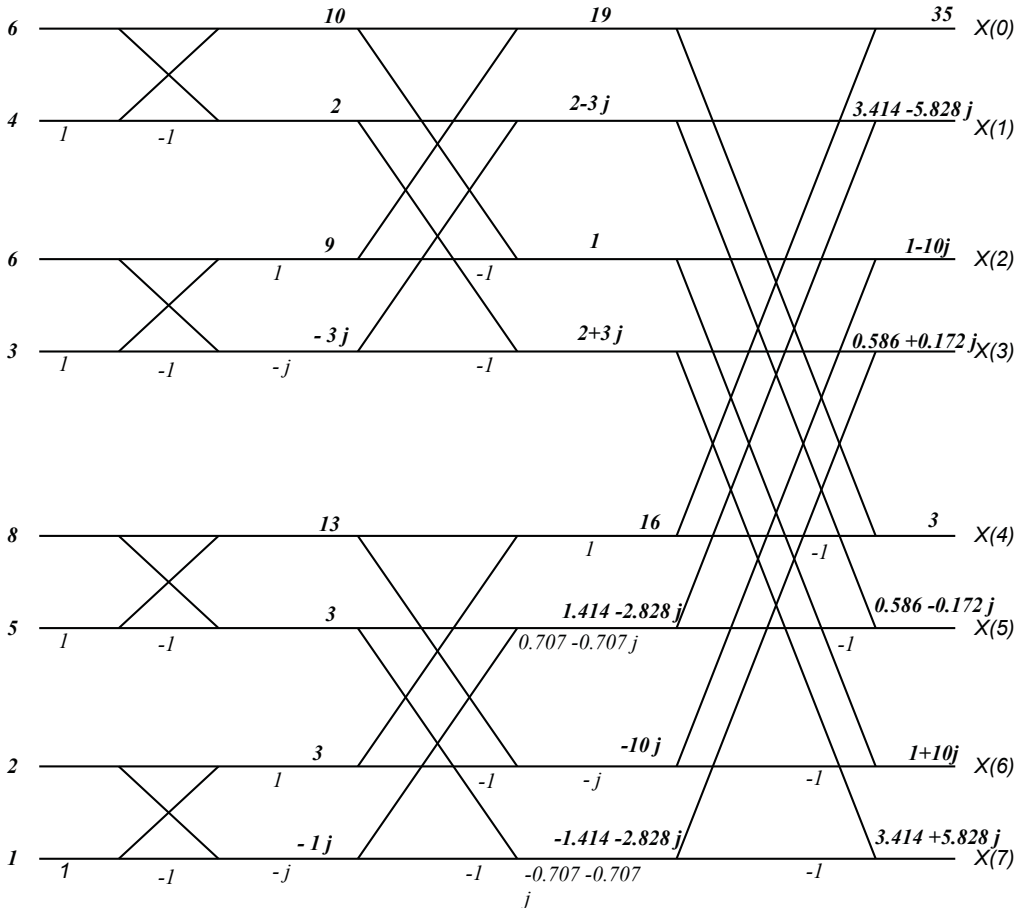
Nilai dari $W^{nk}{}_N$ dapat dihitung dengan $W_N^{nk} = e^{-j\frac{2\pi nk}{N}}$ seperti contoh berikut ini:

$$W_8^1 = e^{-j\frac{2\pi 1}{8}} = e^{-j\frac{1}{4}\pi} = 1\angle -\tfrac{1}{4}\pi = 0.707 - 0.707\,j$$

$$W_8^3 = e^{-j\frac{2\pi 3}{8}} = e^{-j\frac{3}{4}\pi} = 1\angle -\tfrac{3}{4}\pi = -0.707 - 0.707\,j$$

Untuk memahami dengan lebih jelas, maka kita akan menghitung spektrum frekuensi dari *x(n)* sebagai berikut:

x(0) = 6
x(1) = 8
x(2) = 6
x(3) = 2
x(4) = 4
x(5) = 5
x(6) = 3
x(7) = 1



Gambar 4.12
Contoh menghitung FFT dengan diagram *butterfly*

Cara menghitung dengan diagram *butterfly* adalah memperlakukan semua angka di bawah garis sebagai faktor pengali.

- Angka ***10*** pada baris paling atas diperoleh dari ***6*** + (***4*** x ***1***).
- Angka ***2*** dibawahnya adalah ***6*** + (***4*** x ***1*** x ***-1***).
- Angka ***9*** diperoleh dari ***6*** + (***3*** x ***1***),
- sedangkan angka ***-3j*** diperoleh dari [***6*** + (***3*** x ***1*** x ***-1***)] x ***–j***.

Perhitungan ini akan menghasilkan *X(k)* sebagai berikut:

```
X(0)    =    35
X(1)    =    3.414 - 5.828 j
X(2)    =    1 – 10 j
X(3)    =    0.586 + 0.172 j
X(4)    =    3
X(5)    =    0.586 - 0.172 j
X(6)    =    1 + 10 j
X(7)    =    3.414 + 5.828 j
```

Kita dapat melakukan perhitungan FFT dari *x(n)* di atas pada Matlab dengan perintah:

```
>> x=[6 8 6 2 4 5 3 1]';
>> X=fft(x)
```

Anda dapat mempelajari lebih lanjut untuk mengambarkan diagram *butterfly* untuk 16 atau lebih sample.

4.4 Inverse FFT (IFFT)

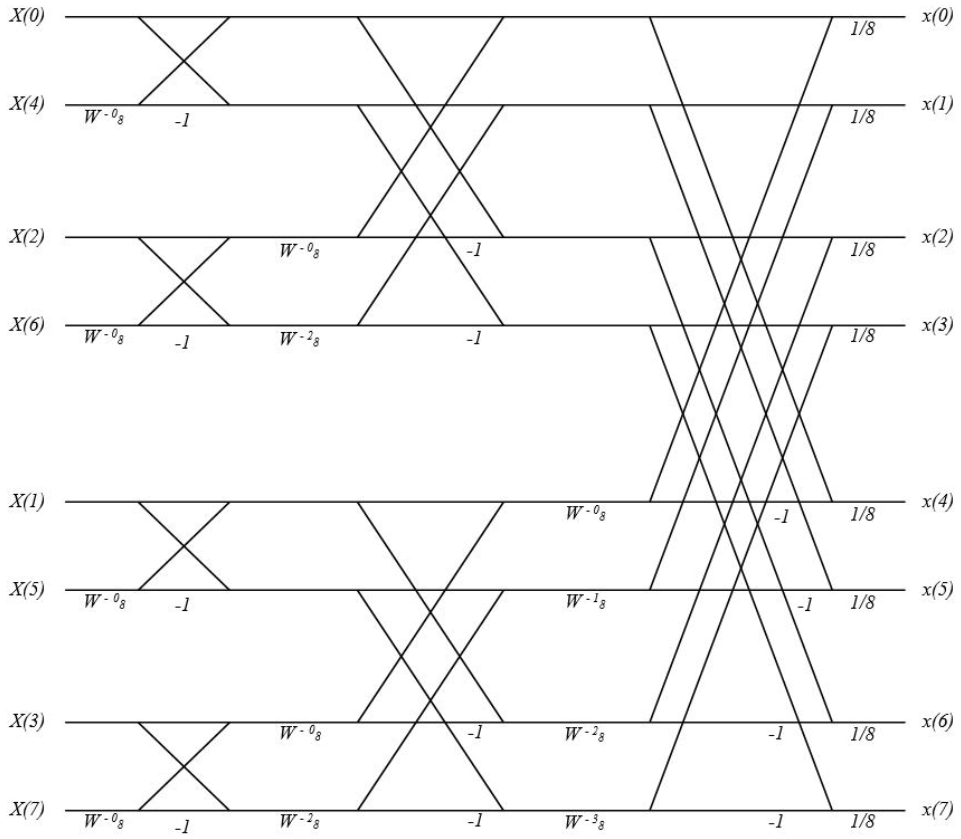
Inverse FFT mentransformasi spektrum frekuensi, *X(k)* kembali menjadi sinyal di domain waktu *x(n)*.

$$x(n) = \frac{1}{N}\sum_{k=0}^{K-1} X(k) e^{+j\frac{2\pi k n}{N}}$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{K-1} X(k) W_N^{-nk}$$

Perbedaan persamaan FFT dan IFFT di atas adalah pada pangkat dari *W* yang menjadi negatif dan hasilnya dikalikan dengan *1/N*. Diagram *butterfly* dapat juga digunakan dengan mengganti pangkat *W* dan menambahkan pengali *1/N*.

Gambar 4.13 menunjukkan diagram *butterfly* dari proses perhitungan IFFT untuk 8 sample.



Gambar 4.13
Diagram *butterfly* IFFT untuk 8 sample

Hasil dari perhitungan IFFT bisa berbentuk bilangan kompleks tetapi yang digunakan hanyalah bagian rilnya saja.

SOAL LATIHAN

1. Dengan menggunakan Matlab, buatlah sebuah sinyal sinusoidal diskrit dengan amplitudo *3*, frekuensi *5 Hz* dari *0* sampai *1* detik. Gunakan frekuensi sampling *15 Hz*. Hitung dan tampilkan spektrum amplitudonya. Tentukan frekuensi dari masing-masing *k*, khususnya *k* yang memiliki amplitudo tertinggi. Apakah spektrum di domain frekuensi menunjukkan amplitudo dan frekuensi sinyal tersebut?

2. Dengan menggunakan Matlab, buatlah sebuah sinyal sinusoidal diskrit dengan amplitudo *2*, frekuensi *250 Hz* dari *0* sampai *0.05* detik. Gunakan frekuensi sampling *5000 Hz*. Hitung dan tampilkan spektrum amplitudonya. Tentukan frekuensi dari masing-masing *k*, khususnya *k* yang memiliki amplitudo tertinggi. Apakah spektrum di domain frekuensi menunjukkan amplitudo dan frekuensi sinyal tersebut? Perhatikan hal-hal yang menyimpang dari teori dan carilah penyebabnya.

3. Ulangi soal nomor 2 dengan menggunakan frekuensi sinyal *1 kHz*.

4. Ulangi soal nomor 2 dengan menggunakan sinyal yang merupakan penjumlahan dari sinyal di nomor *2* dan *3*.

5. Ulangi soal nomor 2 dengan menggunakan frekuensi sinyal *3 kHz* dan fase *0.2 rad*.

6. Dengan menggunakan diagram *butterfly*, hitunglah FFT dari sinyal *y(n)* = [*4 2 7 7 6 5 4 9*]. Ulangi perhitungan ini dengan mengunakan Matlab. Bandingkan kedua hasil perhitungan ini.

7. Hitunglah IFFT dari hasil perhitungan di soal nomor 6. Ulangi perhitungan ini dengan mengunakan Matlab. Bandingkan kedua hasil perhitungan ini.

5
Spektrum Frekuensi

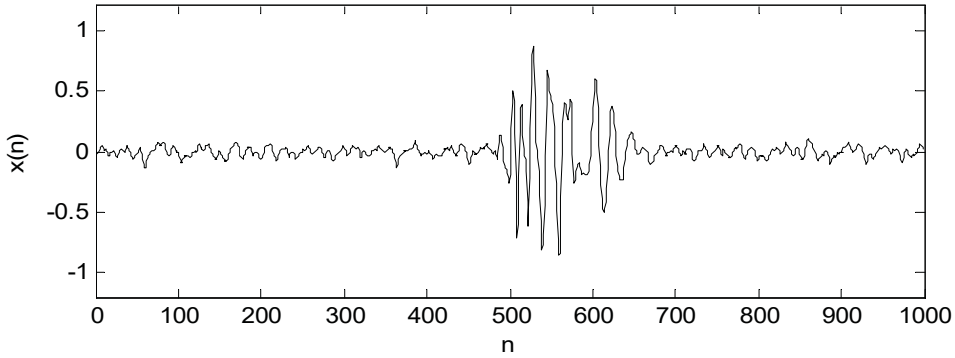
Pada bab ini kita akan menggunakan teknik DFT yang sudah kita pelajari pada Bab 4 dalam menganalisa berbagai sinyal untuk mengetahui komponen sinusoidal atau spektrum frekuensi dari sinyal-sinyal tersebut. Kita akan menghitung DFT dari sinyal dengan jumlah sample yang sangat besar sehingga kita akan menggunakan perintah `fft` pada Matlab. Perintah `fft` pada Matlab menghitung spektrum frekuensi dengan algoritma DFT (bukan FFT) sehingga jumlah sample pada inputnya tidak dibatasi 2^n. Anda dianjurkan untuk membaca *help* dari perintah itu untuk memahami penggunaanya.

Setelah mempelajari materi dalam bab ini maka mahasiswa akan dapat melakukan analisis sinyal berdasarkan komponen spektrum dengan teknik DFT.

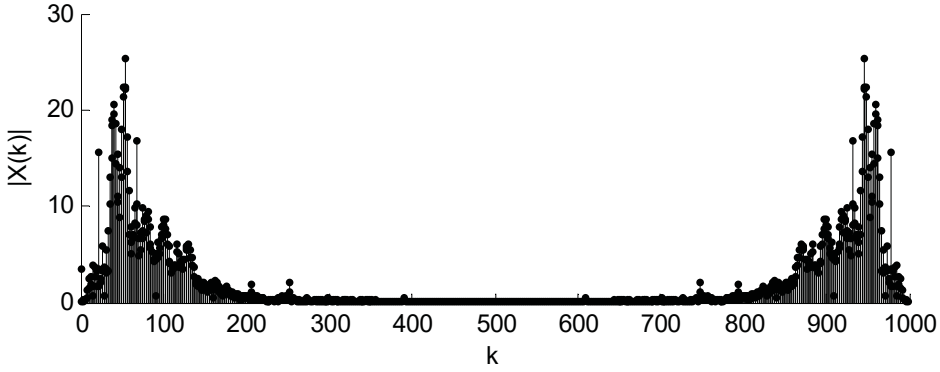
5.1 Spektrum Amplitudo dan Spektrum Daya

Sinyal pada Gambar 5.1 adalah *heart sound* sebanyak *1000* sample dengan $f_s = 2.205\ kHz$. DFT dari sinyal tersebut (Gambar 5.2) menunjukkan penyebaran frekuensi (spektrum frekuensi) dari sample ke-20 sampai sample ke-140 (*44.1 Hz*

sampai *308.7 Hz*), dan adanya komponen frekuensi yang dominan yaitu pada sample ke-23 (*50.7 Hz*), 40 (*88.2 Hz*), 53 (*116.8 Hz*), dan 69 (*152.1 Hz*). Komponen frekuensi pada *50.7 Hz* adalah interferensi dari tegangan jala-jala, sedangkan komponen frekuensi yang lain adalah dari sinyal itu sendiri. Informasi ini dapat digunakan untuk menganalisa sinyal *heart sound* tersebut.



Gambar 5.1
Sinyal heart sound



Gambar 5.2
DFT dari sinyal pada Gambar 5.1

Hasil dari DFT dapat ditampilkan dalam format spektrum amplitudo atau dalam bentuk spektrum daya (*power spectrum*). Spektrum amplitudo memberikan informasi tentang amplitudo dari setiap komponen sinusoidal dalam

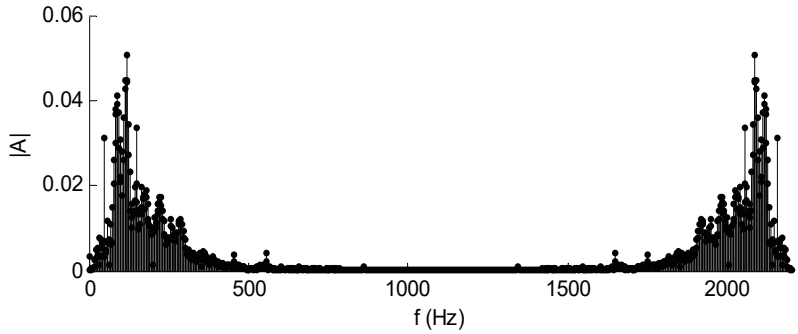
sinyal, sedangkan spektrum daya memberikan informasi tentang daya dari setiap komponen sinusoidal.

Spektrum amplitudo, $|A|$, dari suatu sinyal *x(n)* untuk $0 \leq n \leq N$ adalah

$$| A(k) | = \frac{1}{N+1} X(k) \quad \text{untuk } k = 0$$

$$| A(k) | = \frac{2}{N+1} X(k) \quad \text{untuk } k \text{ selain } 0$$

Gambar 5.3 menunjukkan spektrum amplitudo dari sinyal pada Gambar 5.1.

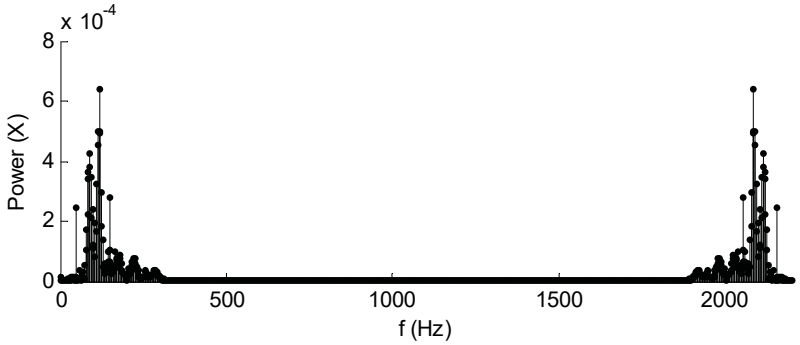


Gambar 5.3
Spektrum Amplitudo dari sinyal pada Gambar 5.1

Spektrum daya, *Power(X)*, dari suatu sinyal *x(n)* untuk $0 \leq n \leq N$ adalah

$$Power\{X(k)\} = \frac{|x(k)|^2}{(N+1)^2}$$

Gambar 5.4 menunjukkan *power spectrum* dari sinyal tersebut.

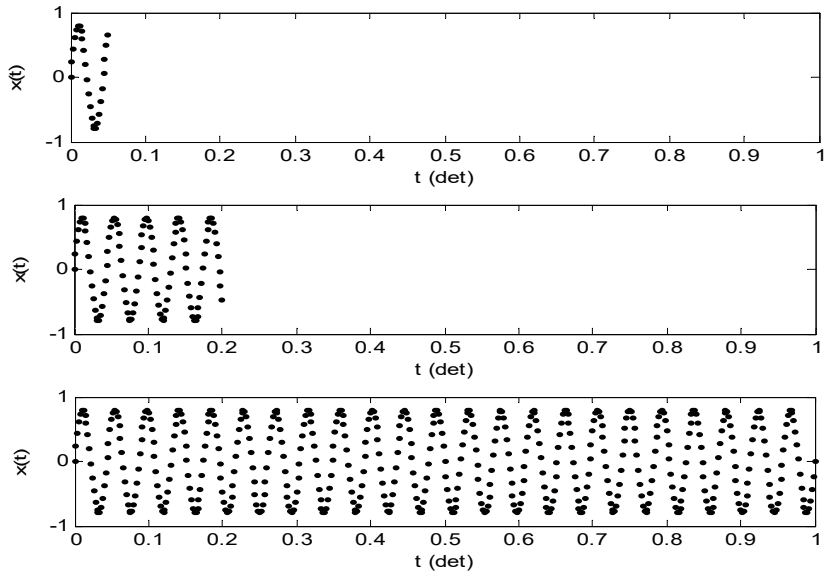


Gambar 5.4
Power spectrum dari sinyal pada Gambar 5.1

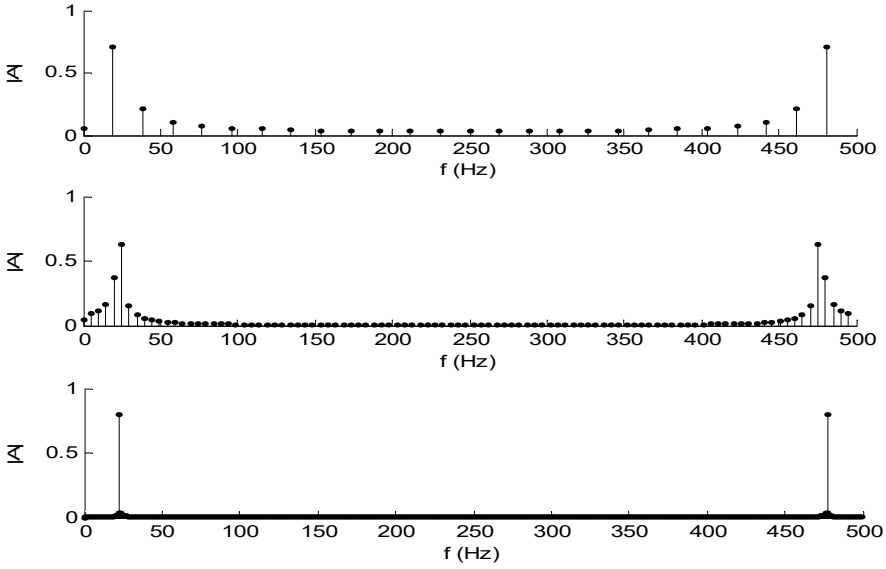
5.2 Panjang sinyal

Ketika menghitung DFT dari suatu sinyal, kita akan dihadapkan pada pertanyaan tentang panjang sinyal yang akan digunakan. Apakah kita perlu menggunakan keseluruhan panjang sinyal yang tersedia. Semakin panjang sinyal yang digunakan, semakin banyak sample yang akan dihitung maka semakin lama pula proses perhitungan DFT. Apa akibatnya jika kita hanya menggunakan sebagian saja dari sinyal tersebut. Berapakah panjang minimum yang harus kita gunakan?

Gambar 5.6 menunjukkan hasil DFT (dalam bentuk spektrum amplitudo) dari sinyal pada Gambar 5.5 (dengan frekuensi *23 Hz* yang disample dengan $f_s = 500\ Hz$) dengan panjang sinyal yang berbeda-beda masing-masing *0.05* detik, *0.2* detik, dan *1* detik. Jika sinyal yang digunakan semakin panjang maka gambar spektrum amplitudo yang dihasilkan semakin akurat. Pada saat panjang sinyal hanya *0.05* detik (Gambar 5.3a) maka hasil FFT-nya (Gambar 5.4a) menunjukkan seolah-olah ada kompenen sinusoidal dengan amplitudo *0.21* pada *38.5 Hz*. Pada saat kita menggunakan sinyal yang lebih panjang maka kita akan tahu bahwa sinyal tersebut tidak memiliki komponen sinusoidal pada *38.5 Hz* melainkan pada *23 Hz* (Gambar 5.4c).

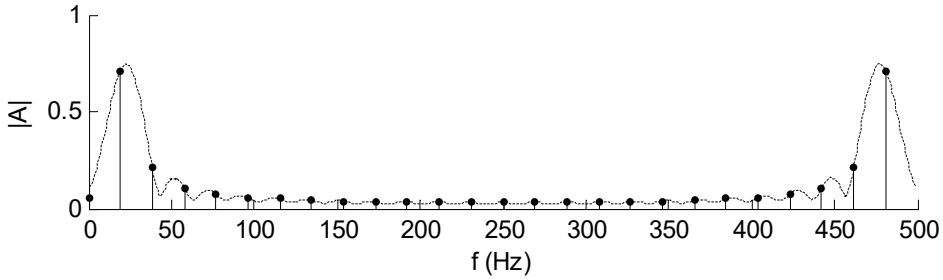


Gambar 5.5
Sinyal *23 Hz* (f_s *= 500 Hz*) dengan panjang yang berbeda
a) sepanjang *0.05* det, b) sepanjang *0.2* det, c) sepanjang *1* det



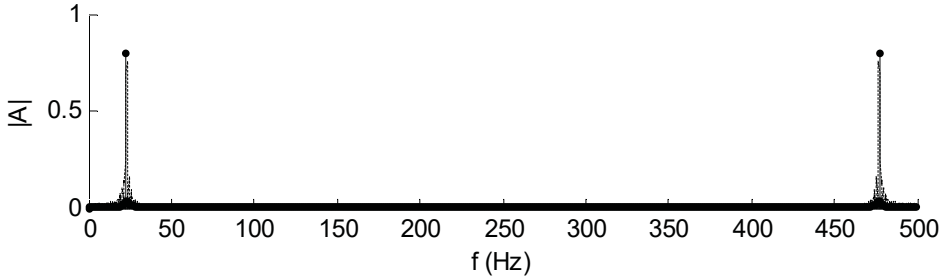
Gambar 5.6
DFT dari sinyal pada Gambar 5.5

Seharusnya sinyal sinusoidal pada Gambar 5.5 hanya memiliki satu komponen frekuensi pada *23 Hz* dengan amplitudo sebesar *0.8*. Munculnya komponen frekuensi palsu pada Gambar 5.6a dan b disebabkan karena sesungguhnya yang digambarkan oleh FFT adalah titik-titik sample dari ***'lobe'*** seperti digambarkan dengan garis putus-putus pada Gambar 5.7.



Gambar 5.7
Main *lobe* dari DFT sinyal dengan panjang 0.05 det.

Semakin pendek sinyal yang digunakan, semakin lebar *main lobe* yang terjadi sehingga gambarnya akan semakin tidak akurat. Gambar 5.8 menunjukkan *main lobe* dari DFT dengan sinyal sepanjang 1 det. Gambar ini menunjukkan bahwa pada sinyal yang panjang, *main lobe*-nya sangat sempit sehingga meredam komponen frekuensi palsu.

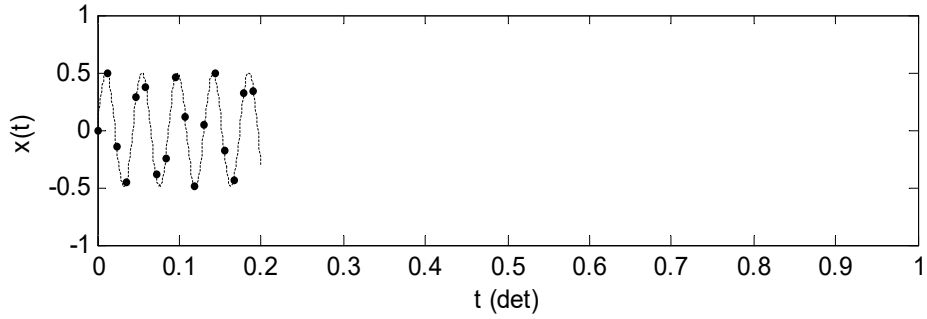


Gambar 5.8
Main lobe dari DFT sinyal dengan panjang 1 det.

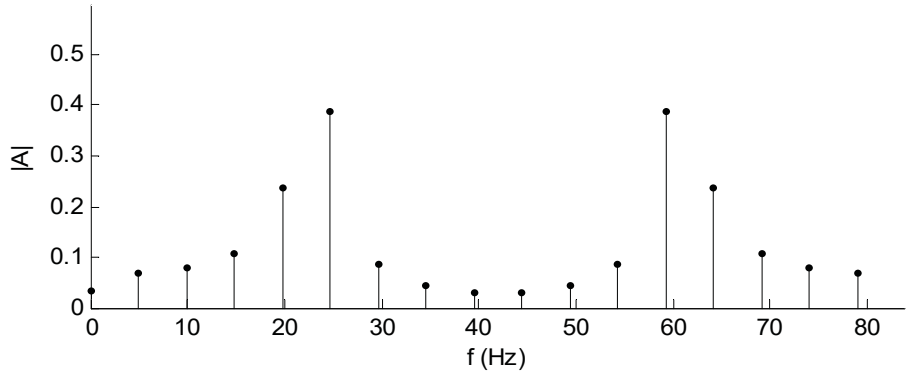
Jika kita ingin menganalisis sinyal dengan komponen sinusoidal yang berdekatan maka kita harus menggunakan sinyal yang panjang agar frekuensi palsu dari *lobe* tidak tumpang tindih dengan komponen sinusoidal dari sinyal.

5.3 Zero Padding

Sebuah sinyal (Gambar 5.9) dengan amplitudo *0.5*, $f = 23\ Hz$ dan $f_s = 84\ Hz$, dianalisis dengan DFT dengan hanya mengambil 17 sample (*0.2 det*). Hasil perhitungan DFT ditunjukkan pada Gambar 5.10 dalam bentuk spektrum amplitudo.

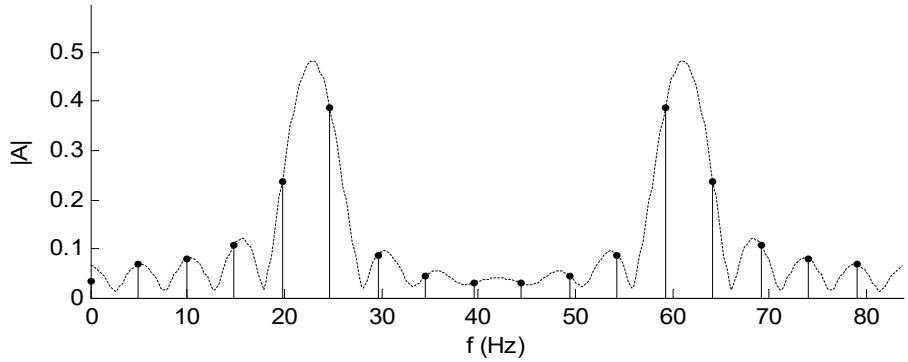


Gambar 5.9
Sinyal dengan 17 sample.



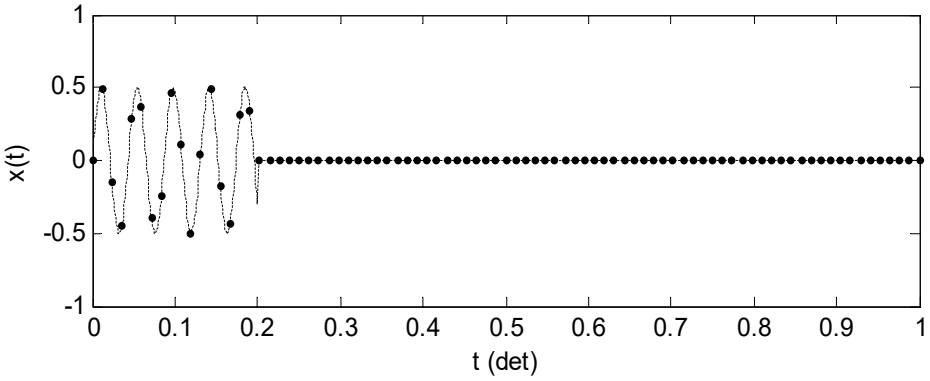
Gambar 5.10
DFT dari sinyal Gambar 5.9

Gambar 5.10 menunjukkan seolah-olah sinyal tersebut mempunyai komponen sinusoidal utama pada sekitar *24 Hz* dengan amplitudo *0.4*. Informasi ini salah karena sinyal tersebut seharusnya memiliki amplitudo *0.5*. Kesalahan ini terjadi karena titik pada gambar hasil DFT tidak terletak di puncak *lobe* seperti ditunjukkan Gambar 5.11. Cara yang paling baik untuk menghindari kesalahan ini adalah dengan memperpanjang sinyal seperti telah dijelaskan pada Sub-Bab 5.2 di atas.

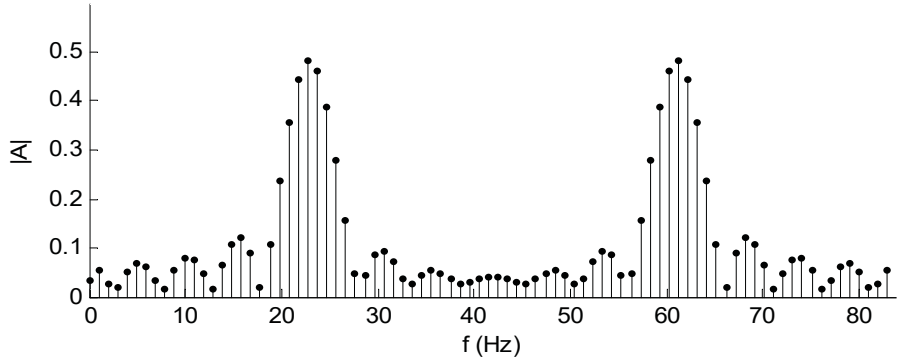


Gambar 5.11
DFT dari sinyal Gambar 5.9 dan gambar main lobe

Jika kita tidak mungkin memperpanjang sinyal tersebut karena berbagai alasan (mis. kita tidak memiliki rekaman lanjutan dari sinyal tersebut) maka kita dapat menggunakan teknik ***zero padding*** yaitu dengan menambahkan banyak sample dengan nilai nol setelah sinyal tersebut (Gambar 5.12).



Gambar 5.12
Zero padding



Gambar 5.13
DFT dari sinyal dengan zero padding

Teknik ***zero padding*** tidak mengubah lebar dari *lobe*, tetapi menambah titik pada gambar spektrum sehingga kita dapat melihat bahwa sinyal tersebut mempunyai amplitudo *0.5* pada frekuensi *23 Hz* (Gambar 5.13).

SOAL LATIHAN

1. Pada direktori Bab 5 dari situs buku ini terdapat beberapa buah sinyal DTMF (dalam format .wav). Sinyal DTMF merupakan sinyal *keypad* telepon berupa gabungan 2 buah frekuensi seperti pada gambar berikut.

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

	Low	High
Busy Signal	480 Hz	620 Hz
Dial Tone	450 Hz	440 Hz

   Gunakan analisis spektrum frekuensi untuk mengetahui nomor yang ditekan pada rekaman itu. Contoh perintah Matlab untuk analisis ini adalah:

```
[a,fs]=wavread('dtmf.wav');
plot(a)
pause;

a1=a(80000:83000);

X=fft(a1);
stem(abs(X),'.');
```

2. Tentukanlah panjang sinyal minimum yang menungkinkan untuk analisis sinyal DTMF di atas. Cobalah dengan berbagai panjang sinyal yang berbeda.

6

Spektrogram

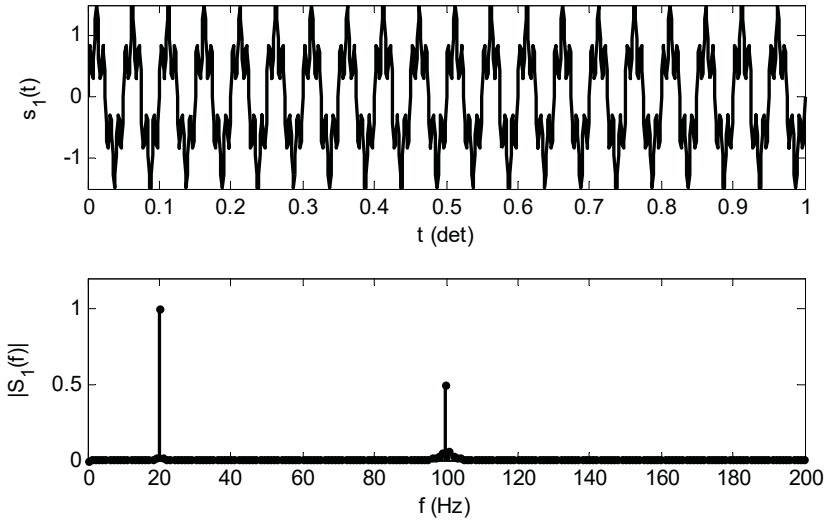
Teknik analisis spektrum frekuensi yang kita pelajari di Bab 5 berasumsi bahwa sinyal input bersifat stasioner. Pada bab ini kita akan mempelajari teknik analisis spektrum frekuensi dengan spektrogram. Spektrogram berguna untuk menganalisisi spektrum frekuensi dari sinyal yang non-stasioner.

Setelah mempelajari materi dalam bab ini maka mahasiswa akan dapat melakukan analisis sinyal non-stasioner dengan spektogram.

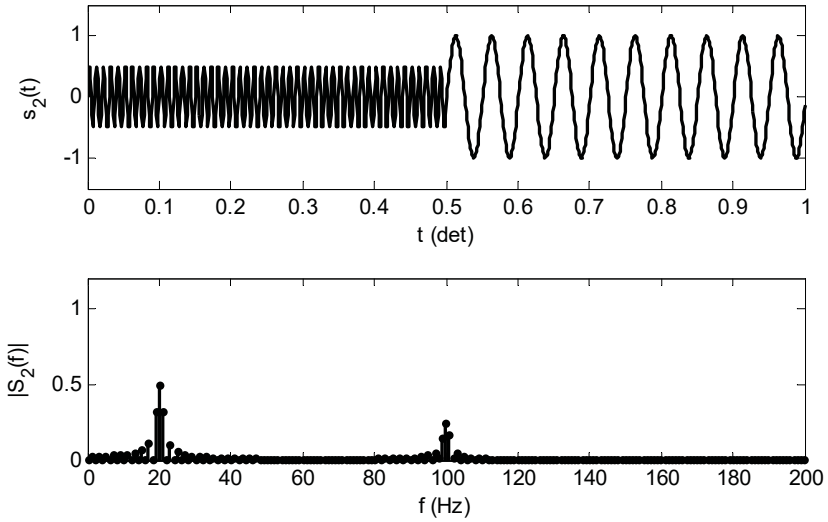
6.1 Sinyal Stasioner dan Non-stasioner

Gambar 6.1 menunjukkan sinyal-sinyal yang memiliki dua komponen sinusoidal. Kedua komponen sinusoidal tersebut bercampur dari $t = 0$ sampai $t = 1$ detik. Sinyal pada Gambar 6.2 juga memiliki dua komponen sinusoidal yang sama tetapi tidak tercampur. Komponen pertama terletak pada $t = 0$ detik sampai $t = 0.5$ detik, sedangkan komponen kedua terletak pada $t > 0.5$ detik sampai $t = 1$ detik. Sinyal pada Gambar 6.1 disebut stasioner karena tidak mengalami perubahan komponen sinusoidal, sedangkan sinyal pada Gambar 6.2

disebut non-stasioner karena mengalami perubahan komponen sinusoidal pada $t = 0.5$ detik.



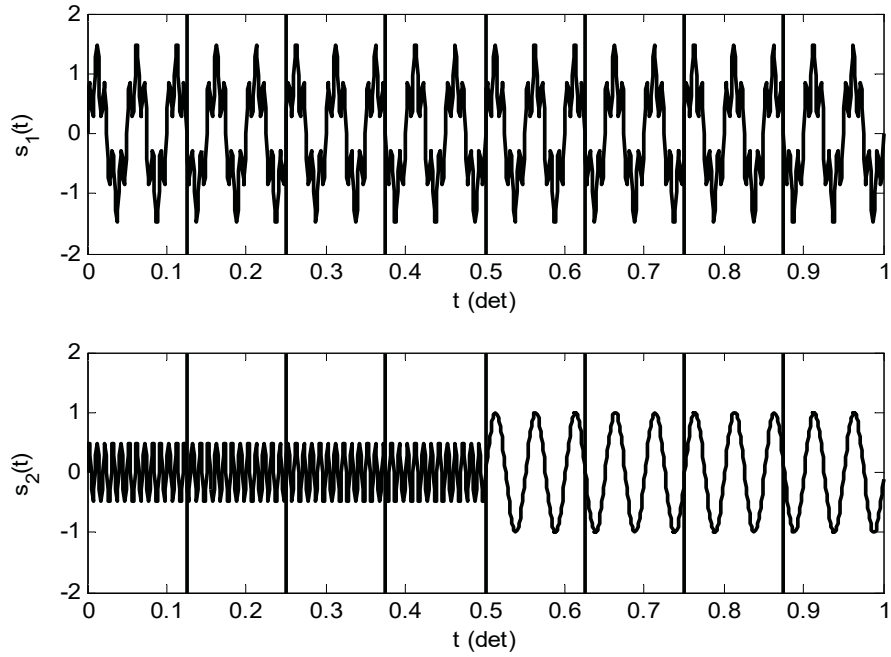
Gambar 6.1
Sinyal $s_1(t)$ dengan dua komponen sinusoidal yang tercampur



Gambar 6.2
Sinyal $s_2(t)$ dengan dua komponen sinusoidal yang terpisah pada $t = 0.5$

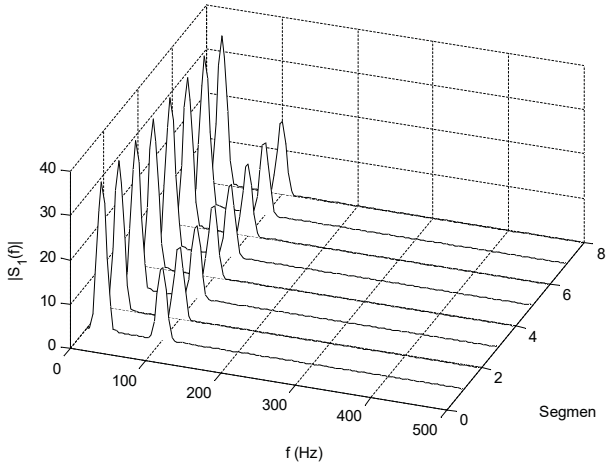
Spektrum frekuensi dari kedua sinyal tersebut menunjukkan adanya dua komponen sinusoidal tetapi tidak memberikan informasi yang cukup tentang apakah kedua komponen sinusoidal tersebut tercampur atau tidak. Jika kita hanya melihat gambar dari spektrum frekuensi, kita tidak dapat menyimpulkan apakah dua komponen sinusoidal tersebut terjadi bersamaan atau terjadi secara bergantian.

Untuk membedakan kedua sinyal di atas maka kita harus memotong sinyal di atas menjadi segmen-segmen kecil dan menghitung spektrum frekuensi dari setiap segmen. Perhitungan seperti ini disebut Short Time Fourier Transform (STFT). Perhitungan ini menghasilkan banyak gambar domain frekuensi, sesuai dengan banyaknya segmen yang dihitung. Sebagai contoh kita akan memotong sinyal di atas menjadi 8 segmen seperti pada Gambar 6.3.

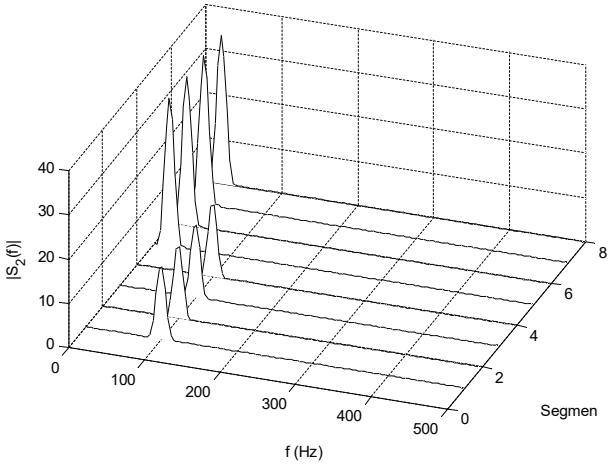


Gambar 6.3
Sinyal $s_1(t)$ dan $s_2(t)$ dipotong menjadi 8 segmen

Hasil perhitungan spektrum frekuensi dari setiap segmen ditunjukkan pada Gambar 6.4 dan 6.5. Gambar-gambar ini menunjukkan bahwa sinyal $s_1(t)$ memiliki dua komponen sinusoidal yang tercampur dari $t = 0$ sampai $t = 1$ detik, sedangan kedua komponen ini terpisah pada sinyal $s_2(t)$.



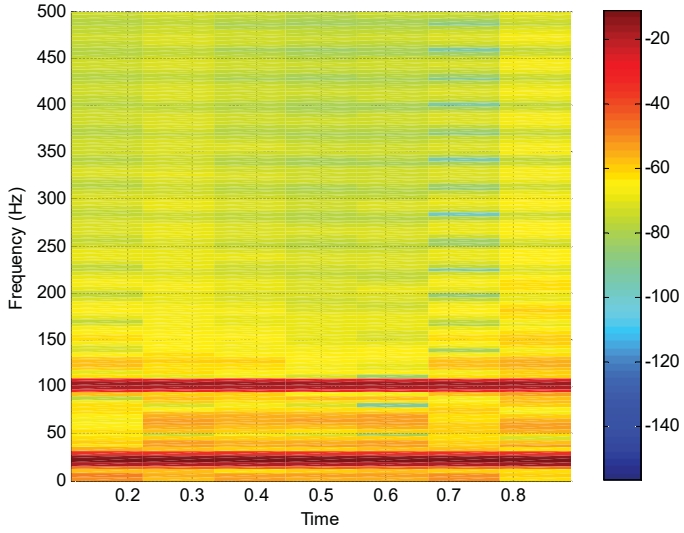
Gambar 6.4
Spektrum frekuensi dari 8 segmen sinyal $s_1(t)$



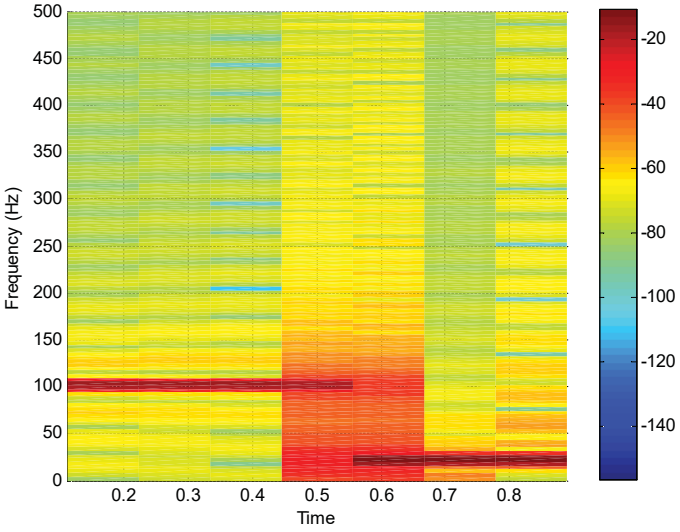
Gambar 6.5
Spektrum frekuensi dari 8 segmen sinyal $s_2(t)$

6.2 Penggambaran Spektrogram

Penggambaran spektrum frekuensi sinyal non-stasioner seperti pada Gambar 6.4 dan 6.5 akan menjadi sangat rumit untuk sinyal yang kompleks dan jumlah segmen yang banyak. Spektrm frekuensi dari sinyal non-stasioner biasa digambarkan dalam bentuk spektrogram seperti pada Gambar 6.6. dan 6.7. Gambar ini sama dengan Gambar 6.4 dan 6.5 yang dilihat dari atas. Puncak yang tinggi deberi warna yang lebih merah. Spektrogram ini menunjukkan dengan jelas bahwa sinyal $s_1(t)$ memiliki dua komponen sinusoidal pada seluruh t sedangkan $s_2(t)$ memiliki dua komponen sinusoidal tetapi tidak terjadi secara besamaan.



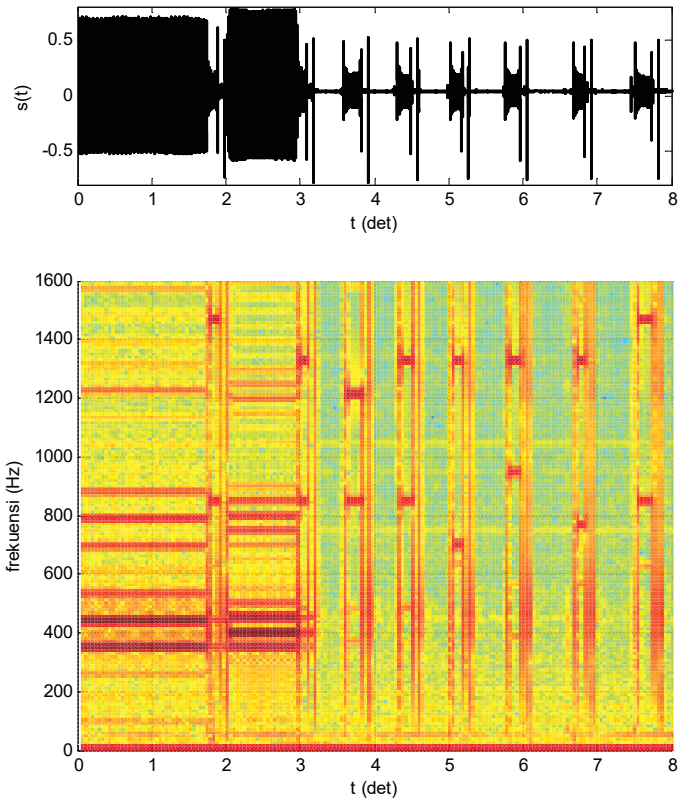
Gambar 6.6
Spektrogram sinyal $s_1(t)$



Gambar 6.7
Spektrogram sinyal $s_2(t)$

Spektrogram dapat dihitung dengan jumlah segmen yang sangat banyak, dan juga dapat dihitung dengan segmen yang saling tumpang tindih (*overlap*)

Gambar 6.8 menunjukkan spektrogram dari rekaman sinyal DTMF. Spektrogram ini dapat dengan jelas menunjukkan keypad yang ditekan pada rekaman ini. Sebagai contoh, bunyi antara $t = 3.5$ *det* sampai $t = 4$ *det*, memiliki komponen frekuensi pada sekitar 850 Hz dan 1200 Hz. Dari data ini kita mengetahui bahwa tombol keypad yang ditekan adalah '7' (lihat tabel di Bab 5).



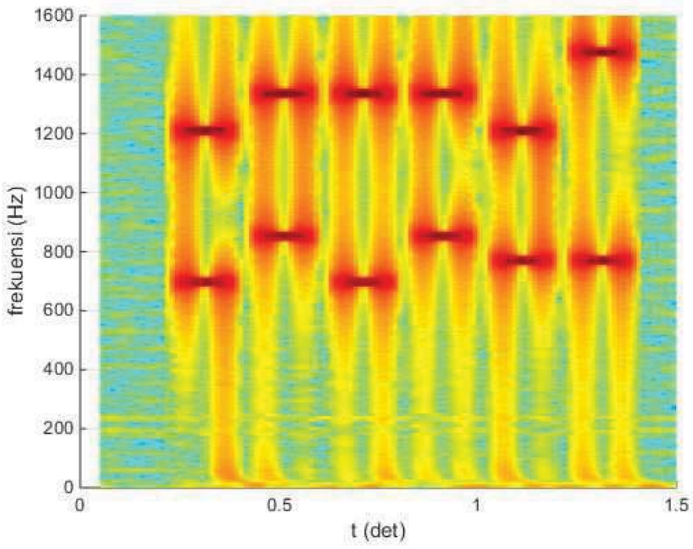
Gambar 6.8
Spektrogram dari sinyal DTMF

6.3 Lebar Window

Sinyal pada Gambar 6.3 dipotong menjadi 8 segmen masing-masing sepanjang 0.2 detik (atau selebar 200 titik sample). Lebar ini disebut *lebar window*. Ketika kita melakukan perhitungan spektrogram, kita harus menentukan lebar window yang digunakan.

SOAL LATIHAN

1. Gambar berikut ini adalah spektrogram dari sebuah sinyal DTMF. Lakukanlah analisis pada sinyal tersebut untuk menentukan tombol yang ditekan pada rekaman tersebut.



2. Lakukanlah analisis spektrogram pada sinyal `ucapan_a.wav` yang merupakan rekaman suara pada saat mengucapkan vokal "A". Tentukanlah frekuensi-frekuensi yang dominan pada rekaman tersebut.

7
Analisis Sinyal Suara

Sejauh ini kita telah mempelajari pengetahuan dasar tentang pemrosesan sinyal digital, khususnya tentang metode untuk menganalisis suatu sinyal berdasarkan spektrum frekuensinya. Pada Bab 6 kita juga telah belajar tentang metode spektrogram untuk melakukan analisis spektrum frekuensi pada sinyal yang non-stasioner seperti sinyal suara.

Pada bab ini, kita akan mengaplikasikan semua pengetahuan dan kompetensi yang sudah kita miliki dari Bab 1 sampai Bab 6 dalam melakukan analisis pada sinyal suara (*speech analysis*). Sinyal suara adalah sinyal non-stasioner yang sering kita temui dalam keseharian kita.

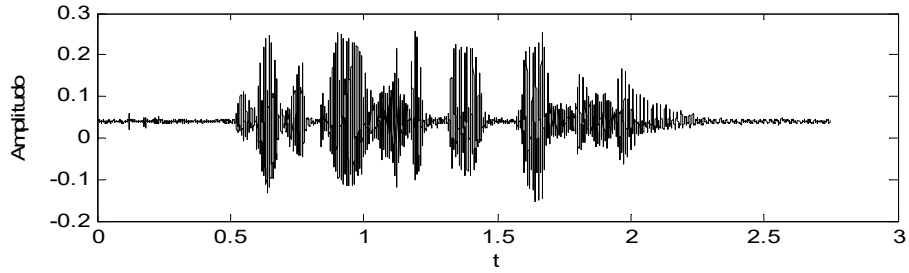
Karena menekankan pada aplikasi dari teori yang sudah dimiliki maka pada bagian akhir dari bab ini akan disajikan panduan praktikum analisis sinyal suara. Sebelum itu, bagian awal dari bab ini akan memberikan landasan pengetahuan tentang sinyal suara dan aspek spektrum dari sinyal suara.

Setelah menyelesaikan bab ini, mahasiswa akan memiliki kemampuan untuk mengimplementasikan dan meng-aplikasikan teknik analisis spektrum pada sinyal suara.

Kemampuan ini juga dapat diaplikasikan pada sinyal non-stasioner yang lainnya.

7.1 Sinyal Suara

Sinyal suara adalah data rekaman grafis dari perubahan tekanan udara yang terjadi karena adanya aktivitas dari organ suara dari manusia. Gambar 7.1 menunjukkan sinyal suara dari suatu ucapan yang direkam selama 2.75 detik.



Gambar 7.1
Sinyal dari ucapan 'digital signal processing'

Sinyal suara ini memiliki aplitudo dan frekuensi yang berubah-ubah selama waktu rekaman tersebut. Sinyal suara ini diucapkan untuk berkomunikasi dari seseorang ke orang yang lain sehingga ada banyak informasi yang terkandung dalam sinyal suara tersebut. Informasi yang ada dalam sinyal tersebut adalah antara lain: ucapan apa yang diucapkan, siapakah orang yang mengucapkan ucapan tersebut, dan juga informasi sekunder lainnya seperti apakah orang tersebut sedang dalam keadaan pilek atau tidak sehat pada saat mengucapkan ucapan tersebut.

Secara matematis, sinyal suara sama dengan semua sinyal yang lainnya yang dapat dinyatakan dalam persamaan amplitudo sebagai fungsi dari waktu, *s(t)*. Walaupun

demikian karena kompleksitas dari sinyal ini maka sulit sekali untuk bisa menuliskan persamaan matematis dari suatu sinyal suara secara akurat.

Pemrosesan sinyal suara atau *speech processing* adalah salah satu aplikasi dari pemrosesan sinyal digital yang bertujuan untuk melakukan proses filter pada sinyal suara dan ekstraksi informasi dari sinyal suara. Analisis sinyal suara dapat diaplikasikan dalam berbagai bidang seperti:

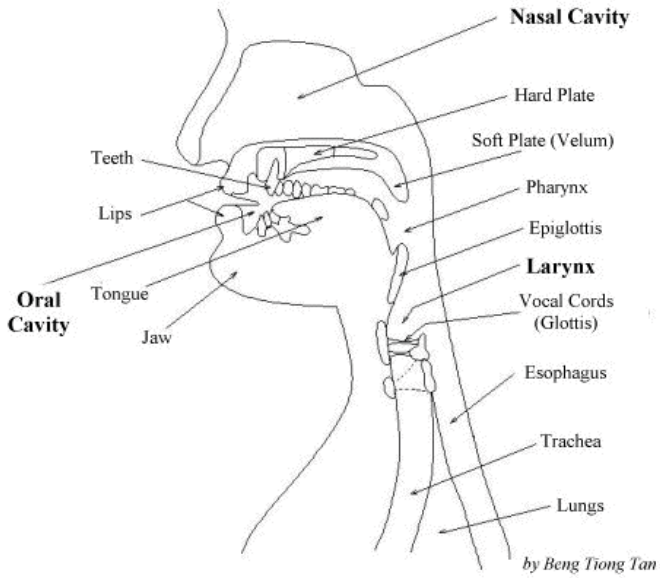
- Interaksi antara manusia dan mesin (Human-Machine Interface)
- Bidang komunikasi digital (Advanced Communication)
- Sekuriti dan personal indentification

Mengaplikasikan teknik pemrosesan sinyal digital dalam analisis sinyal suara bukanlah suatu hal yang mudah. Analisis sinyal suara harus dapat mengatasi berbagai kesulitan yang dihadapi seperti:

- Adanya variasi pada kata (ucapan) yang sama jika diucapkan oleh orang yang berbeda.
- Ucapan yang sama belum tentu akan memiliki bentuk sinyal yang persis sama jika diucapkan ulang, sekalipun oleh orang yang sama.
- Suatu ucapan sangat dipengaruhi oleh intonasi yang digunakan. Intonasi ini ditentukan juga oleh kondisi emosi, kesehatan, dialek, dan penguasaan bahasa dari subyek yang mengucapkan.
- Banyaknya jumlah kosa kata dalam setiap bahasa.
- Adanya variasi hasil rekaman karena pengaruh lingkungan pada saat diucapkan seperti adanya noise suara lingkungan, jarak pembicara ke alat rekam, bahkan kelemahan teknis dari alat perekam.

7.2 Proses Pembangkitan Suara

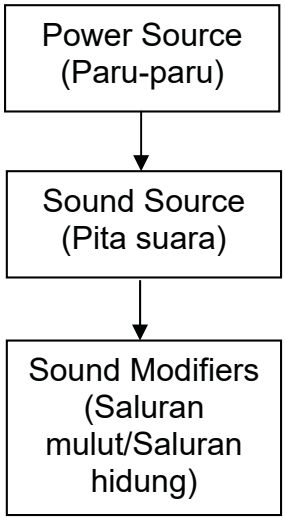
Untuk dapat menganalisis suara dengan baik maka sangat perlu bagi kita untuk memahami proses mekanis dan bilogis dari pembangkitan suara pada manusia. Suara pada manusia dibangkitkan melalui aktivitas dari organ suara seperti terlihat pada Gambar 7.2. Beberapa organ utama adalah paru-paru sebagai pembangkit takanan udara, pita suara sebagai pembangkit frekuensi suara, saluran hidung (*nasal cavity*), saluran mulut (*oral cavity*), serta berbagai organ lain yang berfungsi untuk memodifikasi bunyi suara yang dihasilkan.



Gambar 7.2
Organ Suara pada Manusia

Proses pembangkitan suara pada manusia secara garis besar digambarkan pada Gambar 7.3. Proses ini dimulai dari pembangkitan tekanan udara untuk menciptakan hembusan udara oleh paru-paru. Hembusan udara ini dialirkan melalui *trachea*, pita suara, kemudian melalui saluran mulut (*oral track*) atau saluran hidung (*nasal track*). Pada saat udara

melewati pita suara maka pita tersebut akan bergetar dan mengasilkan suara dengan nada dasar (*sound source*). Suara ini kemudian disalurkan melalui *oral/nasal track* (dengan berbagai komponen di dalamnya) yang berfungsi untuk memodifikasi nada dasar yang dibangkitkan oleh pita suara menjadi berbagai variasi bunyi yang dikenal dengan berbagai fonem (*phoneme*).



Gambar 7.3
Proses Pembangkitan Sinyal Suara

7.3 Voiced dan Unvoiced Speech

Dari sisi pemrosesan sinyal, fonem dalam suatu ucapan (*speech*) dapat dibedakan atas dua kelompok yaitu *voiced speech* (ucapan yang dihasilkan melalui getaran pita suara) dan *unvoiced speech* (ucapan yang dihasilkan tanpa adanya getaran pita suara).

Voiced speech dapat diamati pada ucapan dari semua huruf hidup (*vocal*) seperti "a", "e", "i", "u", dan "o" serta beberapa ucapan lainnya.

Unvoiced speech dapat diamati pada ucapan dari konsonan. Bunyi suara yang dihasilkan pada unvoiced speech dihasilkan oleh getaran atau turbulensi udara di saluran mulut atau saluran hidung. Sebagai contoh, konsonan "m" dan "n" dihasilkan dari bunyi turbulensi udara di saluran hidung (*nasal track*). Konsonan "s" dihasilkan dari bunyi friksi udara yang bertabrakan di sekitar ujung lidah dan gigi atas. Konsonan "p" dihasilkan dari ledakan udara karena bibir yang tadinya tertutup kemudian dibukakan secara mendadak dan diikuti vokal "e". Demikian juga dengan konsonan "b", "t" dan lain sebagainya.

Secara garis besar, *unvoiced speech* dibagi atas:
- frikatif (bunyi yang dihasilkan karena adanya gesekan udara) seperti "s", "f", "h", dan yang bercampur dengan voiced seperti "v", "z",
- stop/eksplosif ("b", "d", "g", "p"),
- afrikatif (tertutup kemudian diikuti olah pembukaan kecil yang menimbulkan turbulensi) seperti "ch" dan "c",
- nasal ("m" dan "n"),
- semivokal ("r" dan "l"),
- dan beberapa bentuk lainnya.

Analisis sinyal suara yang dilakukan pada bab ini hanya difokuskan pada *voiced speech* karena memiliki frekuensi yang dapat dianalisis dengan jelas.

7.4 Frekuensi dari Voiced Speech

Voiced speech memiliki 4 buah frekuensi utama yang membedakan ucapan dari vokal tersebut yaitu pitch (nada dasar) f_0, dan formants f_1, f_2, dan f_3.

Pitch, f_0, adalah getaran dasar dari pita suara yang dapat memberikan informasi jenis kelamin dari orang yang mengucapkan vokal tersebut. Frekuensi f_0 dari laki-laki berkisar antara 85 Hz sampai 180 Hz dengan nilai rata-rata 150 Hz, sedangkan f_0 dari perempuan adalah antara 165 Hz sampai 300 Hz dengan nilai rata-rata 225 Hz. Anak-anak (baik laki-laki maupun perempuan) memiliki range frekuensi f_0 yang lebih tinggi yaitu dengan nilai rata-rata 265 Hz. Batasan frekuensi ini dapat memiliki nilai yang berbeda untuk setiap etnis dan setiap bahasa tetapi masih akan berada di kisaran frekuensi seperti di atas.

Frekuensi formants f_1, f_2, dan f_3 menunjukkan vokal apa yang diucapkan. Secara umum frekuensi formants adalah seperti pada tabel berikut ini.

Tabel 7.1
Frekuensi Formants

	Frekuensi (Hz)		
	f_1	f_2	f_3
a	600	1000	2250
e	400	1600	2400
i	250	1750	2600
u	400	750	2400
o	350	600	2400

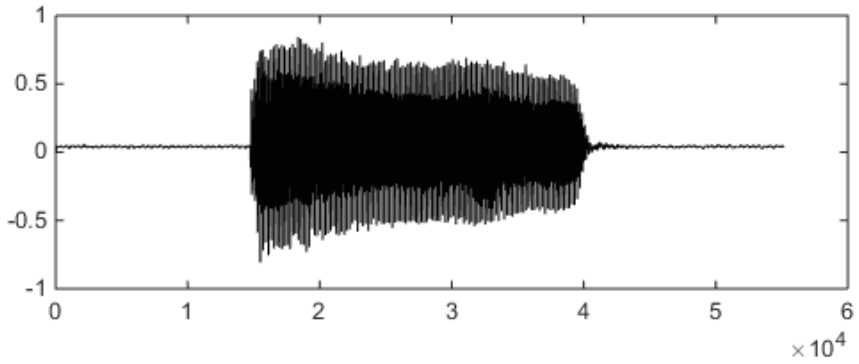
Distribusi frekuensi ini tidak selalu sama untuk setiap orang dan setiap bahasa.

7.5 Praktikum Identifikasi *Pitch*

Untuk mengidentifikasi frekuensi pitch dari suatu *voiced speech*, kita melakukan proses FFT untuk menemukan spektrum frekuensi dari rekaman suara. Pada contoh ini kita akan mengidentifikasi suara ucapan huruf "e" yang direkam dalam file `e.wav` dengan perintah sebagai berikut:

```
[s,fs]=audioread('e.wav');  %membaca dari file
sound(s,fs);
plot(s,'k-')
```

Perintah tersebut membaca file rekaman suara, membunyikannya dan menampilkan gambar dari sinyal suara tersebut seperti di bawah ini.

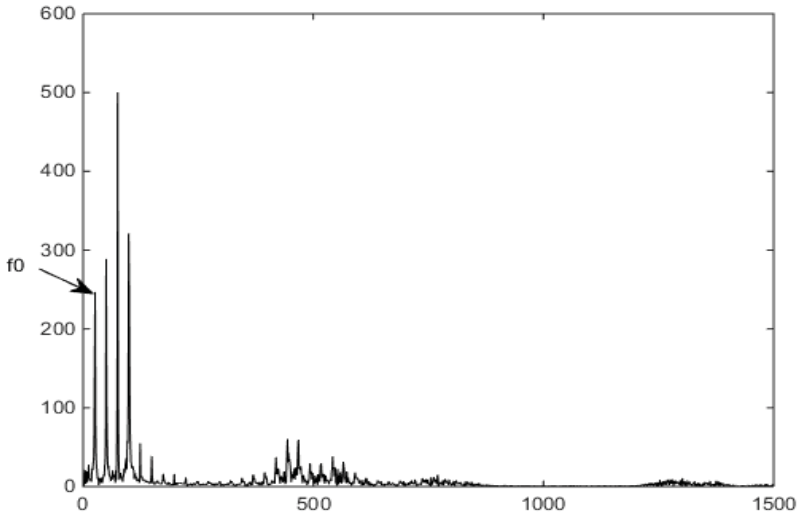


Gambar 7.4
Sinyal Suara e.wav

Karena sinyal suara tersebut baru ada pada titik sampling yang ke 15000 sampai 40000 maka kita akan memotong variabel s agar hanya sinyal suara yang dianalisis. Kita juga menghilangkan *offset* frekuensi dengan mengurangkan rata-rata dari sinyal tersebut. Proses selanjutnya adalah melakukan transformasi Fourier dengan perintah FFT dan menampilkan spektrum frekuensinya.

```
s=s(20000:25000-1)';
s=s-mean(s);
S=fft(s);
plot(abs(S),'k-')
axis([0 1500 0 600]);
```

Frekuensi pitch, f_0, adalah puncak pertama dari spektrum frekuensi seperti ditunjukkan dengan panah pada Gambar 7.5.



Gambar 7.5
Spektrum frekuensi sinyal e.wav

Puncak f_0 tersebut terletak pada titik sample yang k = 25 atau setara frekuensi (sesuai penjelasan di Bab 4)

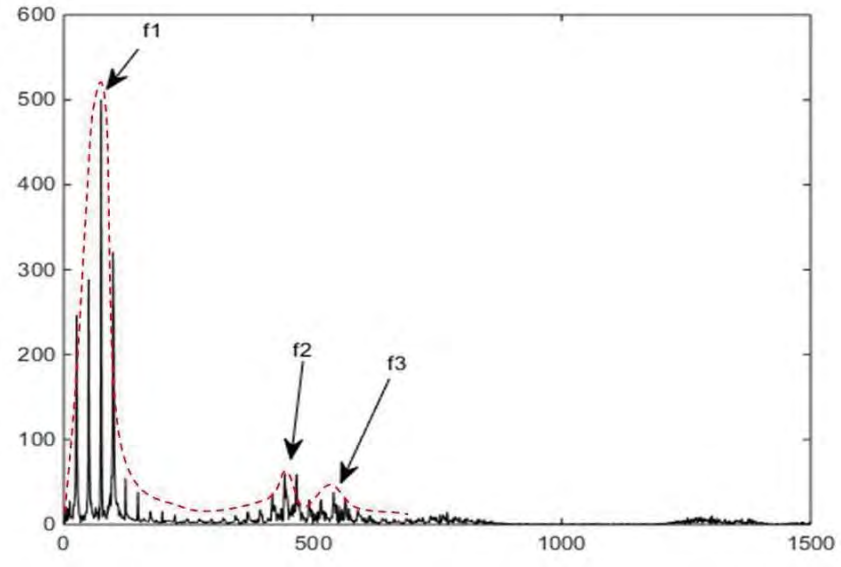
$$fo = \frac{f_s}{jumlah\ sample} k = \frac{22050\ Hz}{5000}(25) = 110.25Hz$$

Nilai pitch ini adalah dalam range suara seorang laki-laki seperti kita bisa konfirmasi dari mendengar rekaman tersebut.

Ulangi langkah praktikum ini untuk file suara lain yang tersedia di direktori (`a.wav, sasa.wav`) dan file lain yang bisa anda rekam sendiri. Lakukan identifikasi f_0 dari suara-suara tersebut dan sesuaikan dengan jenis kelamin dari pengucap pada rekaman tersebut.

7.6 Praktikum Identifikasi Formants, f_1, f_2, dan f_3.

Frekuensi formants f_1, f_2, dan f_3 dapat dilihat pada spektrum frekuensi seperti pada Gambar 7.5, yaitu frekuensi tempat titik puncak dari envelope spektrum frekuensi seperti ditunjukkan pada Gambar 7.6.



Gambar 7.6
Lokasi Formants pada Spektrum frekuensi sinyal e.wav

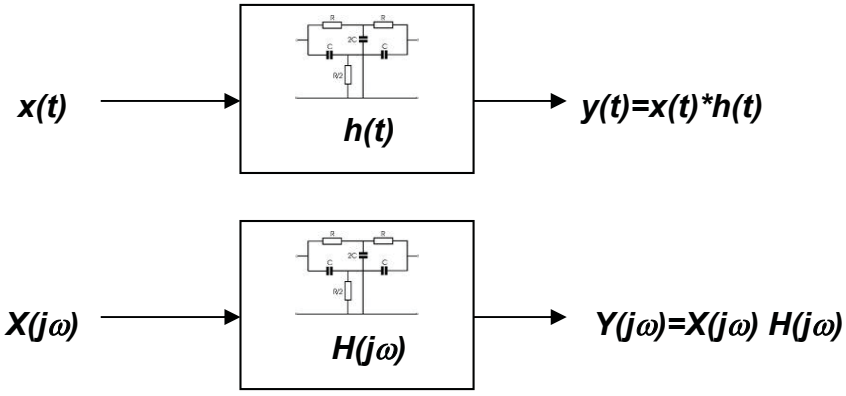
Dari gambar tersebut, f_1, f_2, dan f_3 terletak pada titik sampling ke 75, 444, dan 542 atau setara dengan frekuensi 330 Hz, 1950 Hz, dan 2390 Hz. Jika dilihat pada Tabel 7.1 maka ucapan ini lebih mirip ke vokal "e" sebagaimana terkonfirmasi dari bunyi yang didengar.

Ulangi langkah praktikum ini untuk file suara lain yang tersedia di direktori (`a.wav, sasa.wav`) dan file lain yang bisa anda rekam sendiri. Lakukan identifikasi formants dari suara-suara tersebut dan sesuaikan dengan jenis vokal yang diucapkan.

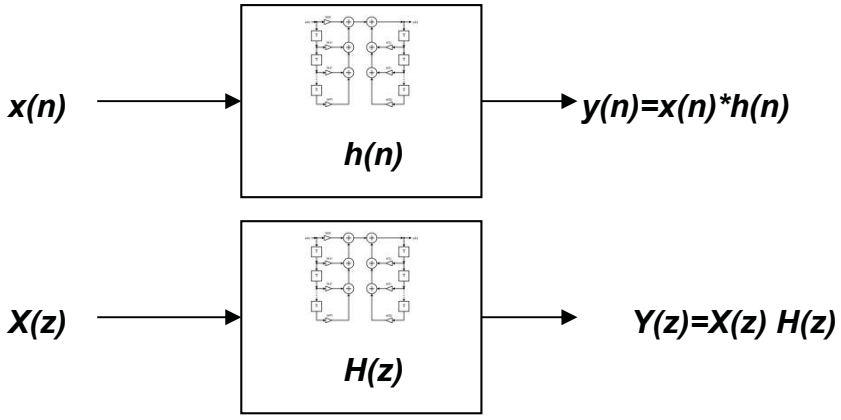
8

Sistem Diskrit

Gambar 8.1 menunjukkan diagram blok dari sistem pemrosesan sinyal analog, sedangkan Gambar 8.2 menunjukkan diagram blok dari sistem pemrosesan sinyal diskrit (*Digital Signal Processing* – DSP).



Gambar 8.1
Sistem Pemrosesan Sinyal Analog



Gambar 8.2
Sistem Pemrosesan Sinyal Diskrit

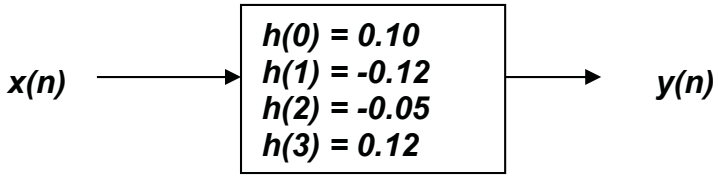
Sistem pemrosesan sinyal analog menerima input sinyal analog dan memprosesnya secara analog dengan komponen-komponen elektronika kemudian menghasilkan sinyal analog sebagai outputnya. Sistem diskrit memproses sinyal diskrit dengan proses aritmetika, karena itu sistem diskrit juga harus dinyatakan secara diskrit.

Sistem analog dinyatakan dengan *h(t)* atau *H(jω)* maka ada dua cara juga untuk menyatakan suatu sistem diskrit yaitu dengan *impulse response, h(n),* atau dengan *transfer function*, *H(z),* seperti pada Gambar 8.2.

Setelah mempelajari bab ini mahasiswa akan dapat mendefinisikan dan membedakan dan mengidentifikasi kelebihan dan kekurangan dari sistem diskrit dengan *impulse response* dan sistem diskrit dengan *transfer function*. Mahasiswa juga akan dapat menghitung output dari sistem diskrit dengan *impulse response* dan sistem diskrit dengan *transfer function* jika diberikan input tertentu.

8.1 Sistem Diskrit dengan Impulse Response

Gambar 8.3 menunjukkan contoh suatu sistem diskrit yang dinyatakan dalam bentuk *impulse response*. Sistem ini adalah sistem dengan orde-4 karena memiliki 4 nilai *h(n)*.



Gambar 8.3
Sistem Diskrit dengan Impulse Response

Output dari sistem ini dihitung dengan menggunakan *linear convolution*:

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

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\,h(n-k)$$

Jika sistem dengan empat buah nilai *h(n)* seperti pada Gambar 8.3 tersebut diberi input dengan 11 buah *x(n)* $\{0 \le n \le 10\}$ maka kita dapat menghitung *y(n)*:

$$
\begin{aligned}
y(0) &= x(0)\ h(0) \\
y(1) &= x(1)\ h(0) + x(0)\ h(1) \\
y(2) &= x(2)\ h(0) + x(1)\ h(1) + x(0)\ h(2) \\
y(3) &= x(3)\ h(0) + x(2)\ h(1) + x(1)\ h(2) + x(0)\ h(3) \\
y(4) &= x(4)\ h(0) + x(3)\ h(1) + x(2)\ h(2) + x(1)\ h(3) \\
y(5) &= x(5)\ h(0) + x(4)\ h(1) + x(3)\ h(2) + x(2)\ h(3) \\
y(6) &= x(6)\ h(0) + x(5)\ h(1) + x(4)\ h(2) + x(3)\ h(3) \\
y(7) &= x(7)\ h(0) + x(6)\ h(1) + x(5)\ h(2) + x(4)\ h(3) \\
y(8) &= x(8)\ h(0) + x(7)\ h(1) + x(6)\ h(2) + x(5)\ h(3) \\
y(9) &= x(9)\ h(0) + x(8)\ h(1) + x(7)\ h(2) + x(6)\ h(3)
\end{aligned}
$$

$$
\begin{aligned}
y(10) &= x(10)\,h(0) + x(9)\,h(1) + x(8)\,h(2) + x(7)\,h(3) \\
y(11) &= x(10)\,h(1) + x(9)\,h(2) + x(8)\,h(3) \\
y(12) &= x(10)\,h(2) + x(9)\,h(3) \\
y(13) &= x(10)\,h(3)
\end{aligned}
$$

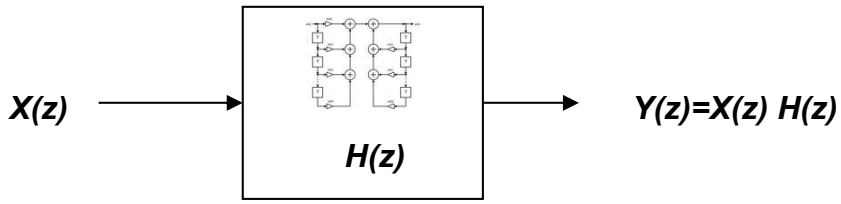
Linear convolution ini menghasilkan 14 data output, *y(n)*. Jumlah data output dari suatu *linear convolution* adalah jumlah data input ditambah jumlah *impulse response* dikurangi *1*.

Secara umum, output dari sistem di atas dapat ditulis:

$$y(n) = (0.10)\,x(n) + (-0.12)\,x(n-1) + (-0.05)\,x(n-2) + (0.12)\,x(n-3)$$

8.2 Sistem Diskrit dengan Transfer Function

Kita telah mengenal sistem diskrit yang dinyatakan dalam format *impulse response*. Pada bagian ini akan dibahas sistem diskrit dalam bentuk *transfer function* seperti pada Gambar 8.4.



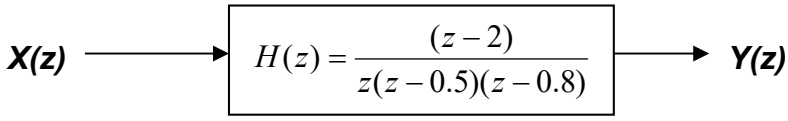
Gambar 8.4
Sistem diskrit dalam bentuk transfer function

Ada dua cara untuk menghitung output dari sistem diskrit dalam format *transfer function*. Cara yang pertama adalah dengan mengalikan *X(z)* dan *H(z)* seperti pada Gambar 8.4, kemudian mentransformasi *Y(z)* menjadi *y(n)*. Cara ini menggunakan transformasi-z yang akan kita pelajari

kemudian. Cara ini tidak umum dipakai karena tidak dapat diterapkan secara real-time. Kita harus mengumpulkan *x(n)* dalam jumlah yang cukup banyak sebelum ditransformasikan menjadi *X(z)*, sehingga ada delay waktu yang cukup besar.

Cara kedua untuk menghitung output dari sistem ini adalah dengan menggunakan persamaan diferensial (*linear constant coefficient difference equation*). Cara kedua ini yang umum diterapkan sehingga bagian ini akan membahas sistem dengan menggunakan persamaan diferensial.

Gambar 8.5 menunjukkan contoh suatu sistem diskrit yang dinyatakan dalam bentuk *transfer function*.



Gambar 8.5
Contoh sistem diskrit dengan transfer function

Untuk menemukan persamaan diferensial dari sistem pada Gambar 7.5 tersebut, kita harus menyederhanakan persamaan *H(z)* sebagai berikut:

$$H(z) = \frac{(z-2)}{z(z-0.5)(z-0.8)}$$

$$H(z) = \frac{(z-2)}{(z^3 - 1.3z^2 + 0.4z)}$$

Sistem ini disebut sistem orde 3 karena pangkat tertinggi dari *z* adalah 3.

$$H(z) = \frac{(z^{-2} - 2z^{-3})}{(1 - 1.3z^{-1} + 0.4z^{-2})}$$

karena $H(z) = \frac{Y(z)}{X(z)}$ maka

$$\frac{Y(z)}{X(z)} = \frac{(z^{-2} - 2z^{-3})}{(1 - 1.3z^{-1} + 0.4z^{-2})}$$

$$Y(z)(1 - 1.3z^{-1} + 0.4z^{-2}) = X(z)(z^{-2} - 2z^{-3})$$

$$Y(z) - 1.3\ Y(z)\ z^{-1} + 0.4\ Y(z)\ z^{-2} = X(z)\ z^{-2} - 2\ X(z)\ z^{-3}$$

kemudian ditransformasi balik ke domain waktu diskrit (domain *n*) dengan transformasi z. Transformasi ini akan dijelaskan lebih detail pada Bab 9, tetapi pada saat yang kita perlukan adalah bentuk transformasi sederhana yaitu:

$$X(n)z^{-a} \quad \Leftrightarrow \quad x(n-a)$$

dan

$$Y(n)z^{-a} \quad \Leftrightarrow \quad y(n-a)$$

sehingga persamaan di atas akan menjadi:

$$y(n) - 1.3\ y(n-1) + 0.4\ y(n-2) = x(n-2) - 2\ x(n-3)$$

Persamaan di atas ini yang disebut *linear constant coefficient difference equation*. Maka kita dapat menghitung *y(n)*:

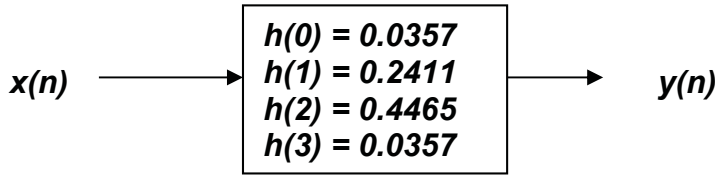
$$y(n) = x(n-2) - 2\ x(n-3) + 1.3\ y(n-1) - 0.4\ y(n-2)$$

Jika sistem seperti pada Gambar 8.5 diberi input dengan 11 buah $x(n)_{\{0 \le n \le N\}}$ maka kita dapat menghitung *y(n)*:

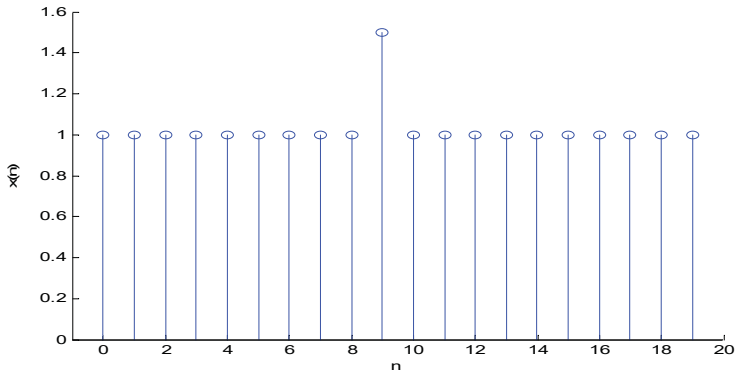
$$
\begin{aligned}
y(0) &= 0 - 0 + 0 - 0 \\
y(1) &= 0 - 0 + 1.3\,y(0) - 0 \\
y(2) &= x(0) - 0 + 1.3\,y(1) - 0.4\,y(0) \\
y(3) &= x(1) - 2\,x(0) + 1.3\,y(2) - 0.4\,y(1) \\
y(4) &= x(2) - 2\,x(1) + 1.3\,y(3) - 0.4\,y(2) \\
y(5) &= x(3) - 2\,x(2) + 1.3\,y(4) - 0.4\,y(3) \\
y(6) &= x(4) - 2\,x(3) + 1.3\,y(5) - 0.4\,y(4) \\
y(7) &= x(5) - 2\,x(4) + 1.3\,y(6) - 0.4\,y(5) \\
y(8) &= x(6) - 2\,x(5) + 1.3\,y(7) - 0.4\,y(6) \\
y(9) &= x(7) - 2\,x(6) + 1.3\,y(8) - 0.4\,y(7) \\
y(10) &= x(8) - 2\,x(7) + 1.3\,y(9) - 0.4\,y(8) \\
y(11) &= x(9) - 2\,x(8) + 1.3\,y(10) - 0.4\,y(9) \\
&\ldots \\
y(n) &= x(n-2) - 2\,x(n-3) + 1.3\,y(n-1) - 0.4\,y(n-2)
\end{aligned}
$$

SOAL LATIHAN

1. Sebuah sistem diskrit dinyatakan dalam bentuk impuse response seperti pada Gambar berikut ini.



   a. Hitunglah (tanpa menggunakan Matlab) output dari sistem tersebut jika diberikan input *x(n)* seperti pada gambar berikut:



   b. Gunakan Matlab (stem) untuk menggambar sinyal diskrit *x(n)*, *h(n)*, dan *y(n)*.
   c. Proses apakah yang dilakukan sistem diskrit tersebut terhadap sinyal input?

2. Sebuah sistem diskrit memiliki impuse response sebagai berikut:

   h(0)=0.0182
   h(1)=0.0488
   h(2)=0.1227
   h(3)=0.1967

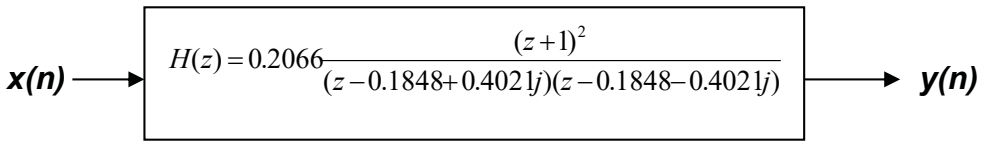
h(4)=0.2273
h(5)=0.1967
h(6)=0.1227
h(7)=0.0488
h(8)=0.0182

Input dari sistem tersebut adalah sinyal *x(n)* yang tersimpan dalam file `sinyal82.mat`. Gunakan program dalam file `modul82.m` untuk menghitung output, *y(n)*. Gunakan Matlab untuk menggambar sinyal diskrit *x(n)*, *h(n)*, dan *y(n)*. Proses apakah yang dilakukan sistem diskrit tersebut terhadap sinyal input?

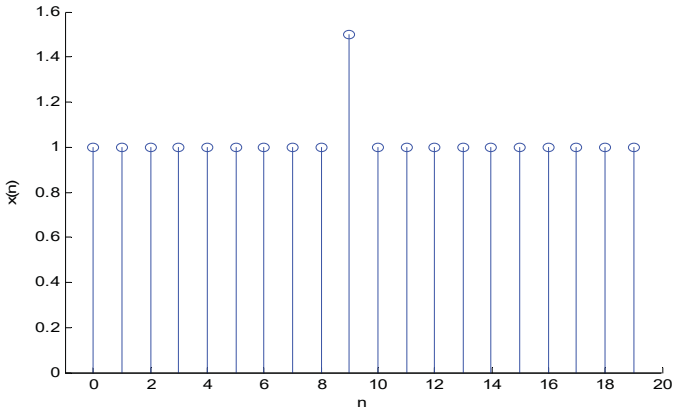
3. Ulangi soal nomor 2 dengan menggunakan sistem yang lain dengan ***h(n)***:

h(0)= 0.0049
h(1)= 0.0103
h(2)= -0.0053
h(3)= -0.0987
h(4)= -0.2463
h(5)= 0.6832
h(6)= -0.2463
h(7)= -0.0987
h(8)= -0.0053
h(9)= 0.0103
h(10)= 0.0049

4. Sebuah sistem diskrit dinyatakan dalam bentuk transfer function seperti pada Gambar berikut ini.



5. Hitunglah (tanpa menggunakan Matlab) output dari sistem tersebut jika diberikan input *x(n)*:



   a. Gunakan Matlab (stem) untuk menggambar sinyal diskrit ***x(n)*** dan ***y(n)***.
   b. Proses apakah yang dilakukan sistem diskrit tersebut terhadap sinyal input?

6. Sebuah sistem diskrit memiliki transfer function sebagai berikut:

$$H(z) = \frac{8.2161e-7\ z^3 + 2.4648e-6\,z^2 + 2.4648e-6\,z + 8.2161e-7}{z^3 - 2.9623\,z^2 + 2.9253101\,z - 0.9630}$$

   Input dari sistem tersebut adalah sinyal *x(n)* yang tersimpan dalam file `sinyal82.mat`. Dengan memodifikasi program dari modul sebelumnya dalam file `modul82.m` untuk menghitung output *y(n)*. Gunakan Matlab untuk menggambar sinyal diskrit *x(n)* dan *y(n)*. Proses apakah yang dilakukan sistem diskrit tersebut terhadap sinyal input?

7. Ulangi soal nomor 6 dengan menggunakan sistem yang lain dengan ***H(z)***:

$$H(z) = 0.4614\,\frac{(z-1)^3}{(z-0.4327)(z-0.5431+0.4417\,j)(z-0.5431-0.4417\,j)}$$

9
Transformasi Z

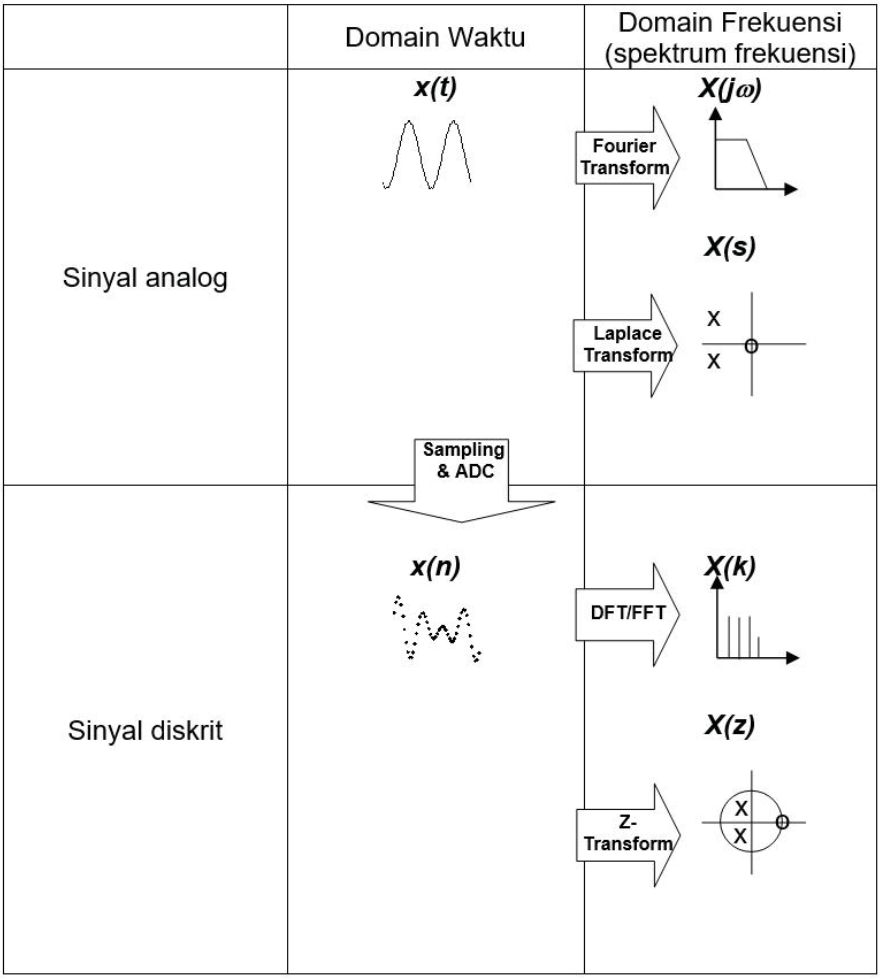
Sampai dengan Bab ke 8, kita telah mempelajari banyak hal tentang pemrosesan sinyal. Kita telah belajar tentang analisis Fourier, domain waktu dan frekuensi, sinyal diskrit, dan sistem dalam bentuk *impulse response* maupun dalam bentuk *transfer function* (persamaan diferensial). Kita sudah tahu bagaimana suatu sistem diskrit bekerja yaitu dengan proses aritmetika dari sample data input.

Sebelum kita masuk ke tahap berikutnya di mana kita akan belajar cara mendesain suatu sistem (menemukan nilai dari *h(n)* atau *H(z)*) maka terlebih dahulu kita akan melihat kembali semua yang sudah kita pelajari. Melihat hubungan di antara hal-hal tersebut dan bagaimana kita dapat berpindah dari satu format bentuk ke bentuk yang lain. Kita akan secara khusus mempelajari hubungan antara *h(n)* dan *H(z)*. Apakah kita dapat menulis *H(z)* dari suatu sistem yang dinyatakan dalam format *h(n)*, atau sebaliknya jika saya memiliki sistem dalam format *H(z)*, apakah saya bisa menemukan *h(n)* dari sistem tersebut?

Kita telah melihat bahwa sinyal dan sistem diperlakukan secara sama. Sinyal dapat ditransformasi ke domain

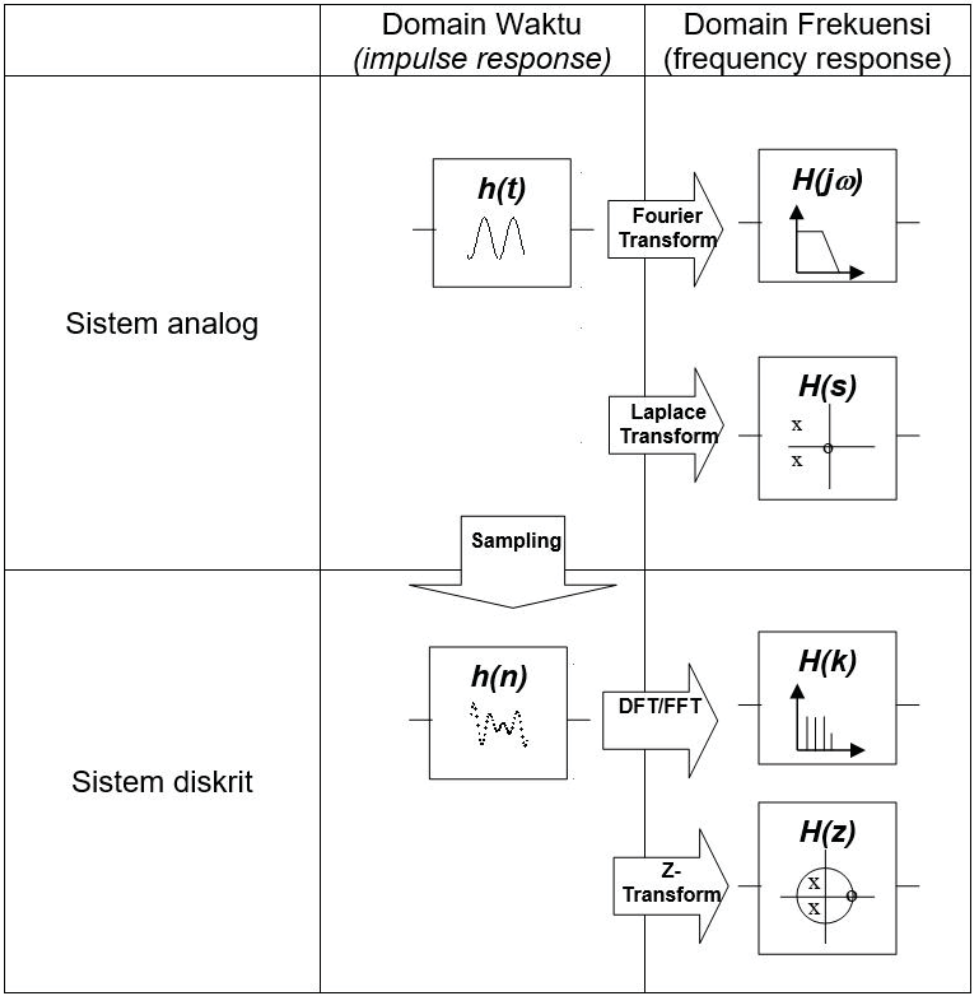
frekuensi, demikian juga dengan sistem. Sinyal dan sistem sama sama ditulis dalam bentuk fungsi matematis, tetapi dengan simbol dan nama yang berbeda. Sinyal biasanya disimbolkan dengan *x(t)* sedangkan sistem disimbolkan dengan *h(t)* dan disebut *impulse response*. Tabel berikut menunjukkan relasi antara sinyal analog dan diskrit dalam domain waktu dan frekuensi.

Tabel 9.1
Relasi Sinyal Analog dan Diskrit



Relasi yang sama juga ditunjukkan pada sistem analog dan sistem diskrit dalam domain waktu dan frekuensi.

Tabel 9.2
Relasi Sistem Analog dan Diskrit



Transformasi Z berfungsi untuk menemukan *transfer function*, *H(z)*, dari sistem diskrit yang dinyatakan dalam bentuk *impulse response*, *h(n)*. Sebaliknya, kita dapat juga

menghitung *h(n)* dari sistem dalam bentuk *H(z)* dengan menggunakan Inverse Z-Transform. Kita dapat juga menemukan *h(n)* dengan cara memberikan sinyal impulse pada bagian input dari suatu sistem diskrit.

Setelah mempelajari bab ini, mahasiswa akan dapat menjelaskan relasi antara *H(z)*, *h(n)*, menghitung dan menggambarkan pole dan zero dari suatu sistem diskrit, menentukan kestabilan sistem, serta dapat melakukan proses transformasi Z dari fungsi-fungsi yang umum dipakai dalam pemrosesan sinyal digital.

9.1 Transformasi Z

Transformasi Z dalam sistem causal dinyatakan dengan persamaan:

$$X(z) = \sum_{n=0}^{\infty} x(n) z^{-n}$$

dimana *z* adalah variabel kompleks. Tabel 9.3 berisi beberapa sifat penting dari transformasi Z.

Tabel 9.3
Sifat Transformasi Z

Sifat	*x(n)*	*X(z)*
Linier	$A\ x_1(n) + B\ x_2(n)$	$A\ X_1(z) + B\ X_2(n)$
Konvolusi	$x(n) * h(n)$	$X(z)\ H(z)$
Time shift	$x(n-a)$	$X(z)\ z^{-a}$

Sifat-sifat di atas yang telah kita gunakan di Modul 7 ketika kita mengubah *H(z)* menjadi persamaan differensial. Transformasi Z dapat dihitung dengan persamaan di atas,

namun untuk memudahkan proses ini pada umumnya digunakan tabel pasangan transformasi Z seperti ditunjukkan pada Tabel 9.4.

Tabel 9.4
Pasangan Transformasi Z

x(n)	***X(z)***	***Daerah Konvergensi***
$\delta(n)$	1	$\|z\|>0$
$\delta(n-a)$	z^{-a}	$\|z\|>0$
$A\,u(n)$	$\frac{Az}{(z-1)}$	$\|z\|>1$
$n\,u(n)$	$\frac{z}{(z-1)^2}$	$\|z\|>1$
$n^2\,u(n)$	$\frac{z(z+1)}{(z-1)^3}$	$\|z\|>1$
$a^n\,u(n)$	$\frac{z}{(z-a)}$	$\|z\|>\|a\|$
$n\,a^n\,u(n)$	$\frac{a\,z}{(z-a)^2}$	$\|z\|>\|a\|$
$e^{-na}\,u(n)$	$\frac{z}{(z-e^{-a})}$	$\|z\|>e^{-a}$
$\sin(an)\,u(n)$	$\frac{z\sin(a)}{z^2-2z\cos(a)+1}$	$\|z\|>1$
$\cos(an)\,u(n)$	$\frac{z(z-\cos(a))}{z^2-2z\cos(a)+1}$	$\|z\|>1$

Transfer function, *H(z)*, dapat ditulis dalam beberapa format yaitu:

- Format polinomial

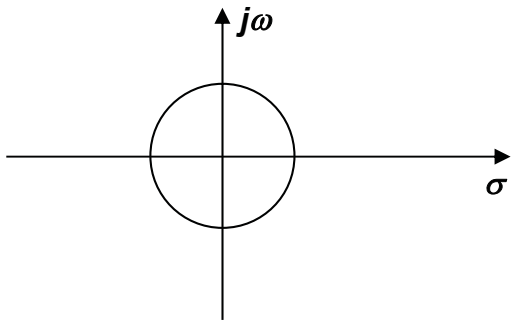
$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + ... + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + ... + a_N z^{-N}}$$

- Format pole-zero

$$H(z) = k \frac{(z - c_1)(z - c_2)...}{(z - d_1)(z - d_2)...}$$

Semua nilai *z* yang menyebabkan *H(z)* bernilai nol disebut ***'zero'*** sedangkan semua nilai *z* yang menyebabkan *H(z)* bernilai ∞ disebut ***'pole'***. Persamaan di atas memiliki zero pada z = c_1 dan z = c_2 dan memiliki pole pada z = d_1 dan z = d_2.

Pole dan zero dari suatu *transfer function* dapat digambarkan dalam bidang Z dimana pole digambarkan dengan 'x' dan zero digambarkan dengan 'o'. Bidang Z digambarkan sebagai bidang kompleks dengan sebuah lingkaran berjari-jari satu berpusat di (0,0).



Gambar 9.1
Bidang Z

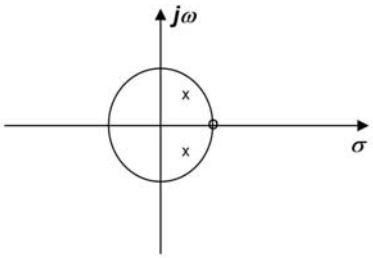
Selama semua pole dari suatu sistem terletak di dalam lingkaran tersebut maka sistem tersebut stabil. Misalnya suatu sistem dengan transfer function:

$$H(z) = \frac{(z-1)}{(z^2 - z + 0.5)}$$

bisa disederhanakan menjadi

$$H(z) = \frac{(z-1)}{(z-0.5+0.5j)(z-0.5-0.5j)}$$

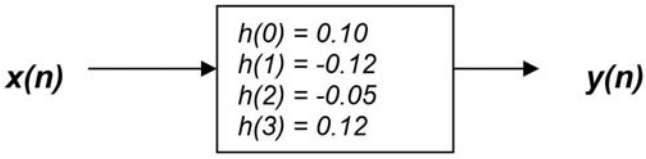
maka sistem ini memiliki zero pada $z = 1$, dan pole pada $z = 0.5 + 0.5j$ dan $z = 0.5 - 0.5j$ seperti pada Gambar 9.2. Sistem ini adalah sistem yang stabil.



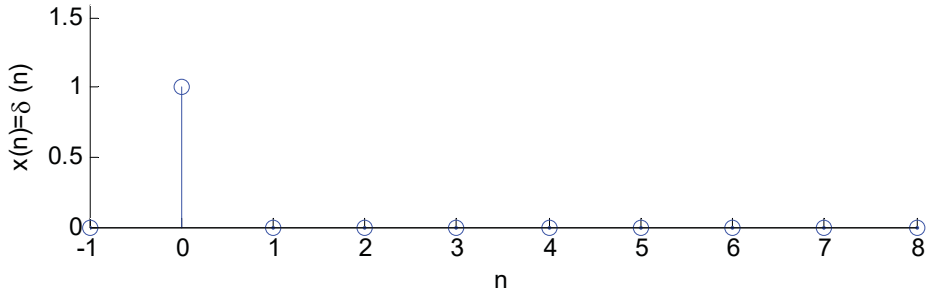
Gambar 9.2
Contoh pole dan zero

9.2 Menemukan Impulse Response

Misalnya kita memiliki suatu sistem seperti pada Gambar 9.3.



Gambar 9.3
Sistem dengan *h(n)*



Gambar 9.4
Sinyal input *x(n) = δ(n)*

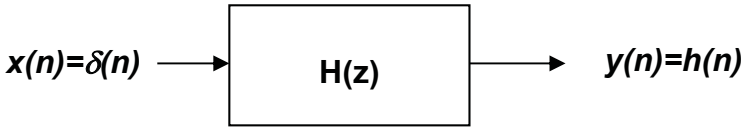
Jika input pada sistem ini adalah sinyal impulse $x(n) = \delta(n)$ seperti pada Gambar 9.4, maka outputnya adalah:

$$
\begin{aligned}
y(0) &= x(0)h(0) = 0.10 \\
y(1) &= x(1)h(0) + x(0)h(1) = -0.12 \\
y(2) &= x(2)h(0) + x(1)h(1) + x(0)h(2) = -0.05 \\
y(3) &= x(3)h(0) + x(2)h(1) + x(1)h(2) + x(0)h(3) = 0.12 \\
y(4) &= x(4)h(0) + x(3)h(1) + x(2)h(2) + x(1)h(3) = 0 \\
y(5) &= x(5)h(0) + x(4)h(1) + x(3)h(2) + x(2)h(3) = 0 \\
y(6) &= x(6)h(0) + x(5)h(1) + x(4)h(2) + x(3)h(3) = 0 \\
y(7) &= x(7)h(0) + x(6)h(1) + x(5)h(2) + x(4)h(3) = 0 \\
y(8) &= x(8)h(0) + x(7)h(1) + x(6)h(2) + x(5)h(3) = 0 \\
&\ldots
\end{aligned}
$$

Dari perhitungan ini dapat dilihat bahwa output, *y(n)*, tepat sama dengan *h(n)* jika menerima input berupa sinyal inpulse. Dengan kata lain, kita dapat mengetahui impulse response, *h(n)*, dari suatu sistem dengan cara memberikan sinyal impulse sebagai inputnya. Inilah sebabnya mengapa *h(n)* disebut *'impulse response'* karena itu merupakan respon sistem ketika diberi impulse.

Jadi jika kita memiliki sistem dalam bentuk *H(z)* maka kita dapat menemukan *h(n)* dari sistem tersebut dengan cara menemukan persamaan diferensialnya kemudian memberikan

$\delta(n)$ pada bagian inputnya seperti diilustrasikan pada Gambar 9.5.



Gambar 9.5
Respon sistem terhadap impulse

Cara lain untuk menemukan *h(n)* dari sistem yang dinyataan dalam bentuk *H(z)* adalah dengan menggunakan inverse Z-transform. Namun cara ini sangat matematis.

9.3 Menemukan Transfer Function

Kita dapat menggunakan transformasi Z untuk menemukan *H(z)* dari suatu sistem yang diketahui dalam bentuk *h(n)*. Ada dua cara yaitu dengan inverse Z-transform dari persamaan konvolusi dan inverse Z-transform dari persamaan *h(n)*.

Cara 1:
Output dari sistem seperti pada Gambar 9.3 dapat dihitung dengan konvolusi

$$y(n) = (0.10)\,x(n) + (-0.12)\,x(n-1) + (-0.05)\,x(n-2) + (0.12)\,x(n-3)$$

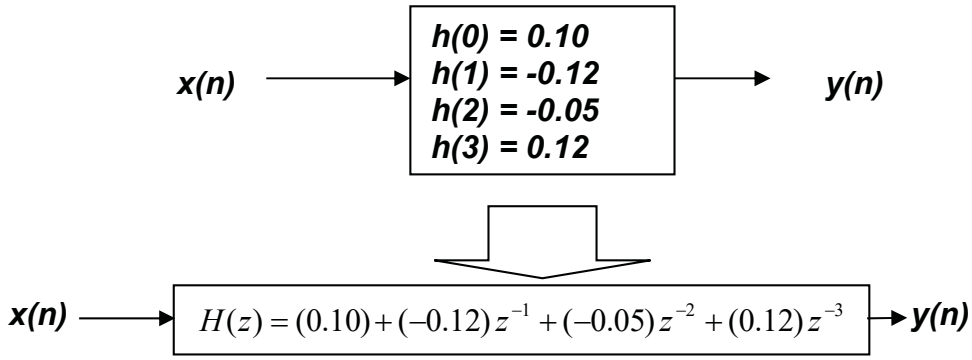
Jika persamaan ini ditransformasi dengan inverse Z-transform maka menjadi

$$Y(z) = (0.10)\,X(z) + (-0.12)\,X(z)\,z^{-1} + (-0.05)\,X(z)\,z^{-2} + (0.12)\,X(z)\,z^{-3}$$

$$Y(z) = X(z)\left[(0.10) + (-0.12)\,z^{-1} + (-0.05)\,z^{-2} + (0.12)\,z^{-3}\right]$$

$$\frac{Y(z)}{X(z)} = (0.10) + (-0.12)\,z^{-1} + (-0.05)\,z^{-2} + (0.12)\,z^{-3}$$

maka $H(z) = (0.10) + (-0.12)\,z^{-1} + (-0.05)\,z^{-2} + (0.12)\,z^{-3}$ seperti diilustrasikan pada Gambar 9.6.



Gambar 9.6
Transformasi *h(n)* menjadi *H(z)*

Cara 2:
Cara lain adalah dengan menulis persamaan *h(n)* dari Gambar 9.3

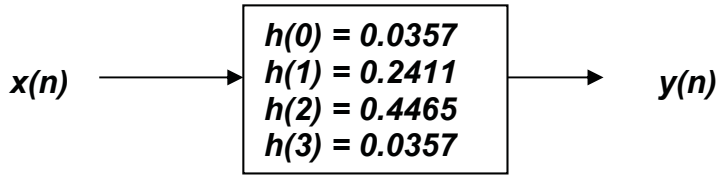
$$h(n) = (0.10)\,\delta(n) + (-0.12)\,\delta(n-1) + (-0.05)\,\delta(n-2) + (0.12)\,\delta(n-3)$$

kemudian mencari *H(z)* dengan transformasi Z seperti pada Tabel 9.4.

$$H(z) = (0.10) + (-0.12)\,z^{-1} + (-0.05)\,z^{-2} + (0.12)\,z^{-3}$$

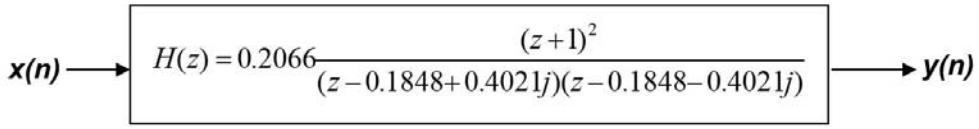
SOAL LATIHAN

1. Sebuah sistem diskrit dinyatakan dalam bentuk impuse response seperti pada Gambar berikut ini.



   a. Temukanlah transfer function, *H(z)*, dari sistem tersebut. Tulislah dalam bentuk pole-zero.
   b. Gambarlah pole dan zero dari sistem tersebut dalam bidang Z. Anda dapat menggunakan perintah 'roots' di Matlab untuk faktorisasi polinomial. Apakah sistem tersebut stabil?

2. Sebuah sistem diskrit dinyatakan dalam bentuk *transfer function* seperti pada gambar berikut ini.



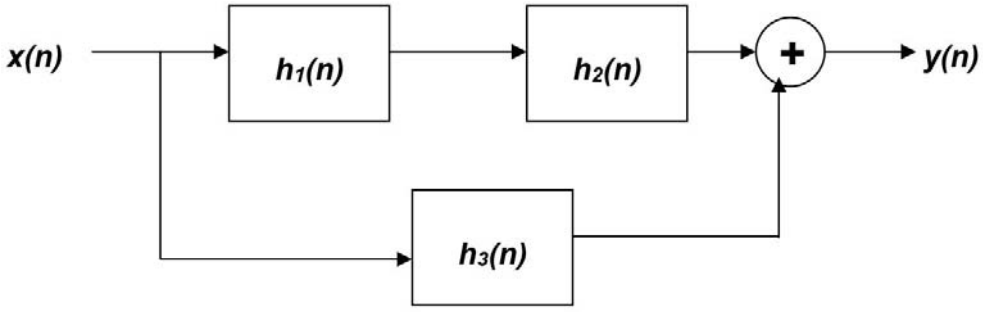
   a. Temukanlah impulse response, *h(n)*, dari sistem tersebut. Anda dapat menggunakan modifikasi dari perintah dalam file modul62.m. Tulislah persamaan *h(n)*.
   b. Gambarlah pole dan zero dari sistem tersebut dalam bidang Z. Apakah sistem tersebut stabil?
   c. Bandingkan impulse response di soal nomor 1, dengan nomor 2. Bila mana kita perlu menyatakan sistem dalam bentuk transfer function, dan bila mana dalam bentuk impulse response.

3. Sebuah sistem diskrit memiliki *transfer function* sebagai berikut:

$$H(z) = \frac{8.2161e-7\ z^3 + 2.4648e-6\,z^2 + 2.4648e-6\,z + 8.2161e-7}{z^3 - 2.9623\,z^2 + 2.9253101\,z - 0.9630}$$

   a. Temukanlah impulse response, *h(n)*, dari sistem tersebut. Anda dapat menggunakan modifikasi dari perintah dalam file `modul82.m`. Tulislah persamaan *h(n)*.
   b. Gambarlah pole dan zero dari sistem tersebut dalam bidang Z. Apakah sistem tersebut stabil?

4. Sebuah sistem diskrit terbentuk dari beberapa sub-sistem:



dimana

$$h_1(n) = (0.0688)\,\delta(n) + (0.8624)\,\delta(n-1) + (0.0688)\,\delta(n-2)$$

$$h_2(n) = (-0.0016)\,\delta(n) + (0.9967)\,\delta(n-1) + (-0.0016)\,\delta(n-2)$$

$$h_3(n) = (-0.0047)\,\delta(n) + (-0.0324)\,\delta(n-1) + (0.9447)\,\delta(n-2) + (-0.0324)\,\delta(n-3) + (-0.0047)\,\delta(n-4)$$

   a. Temukanlah impulse response, *h(n)*, dari sistem tersebut. Anda dapat menggunakan modifikasi dari perintah dalam file `modul82.m`.
   b. Gambarlah pole dan zero dari sistem tersebut dalam bidang Z. Apakah sistem tersebut stabil?

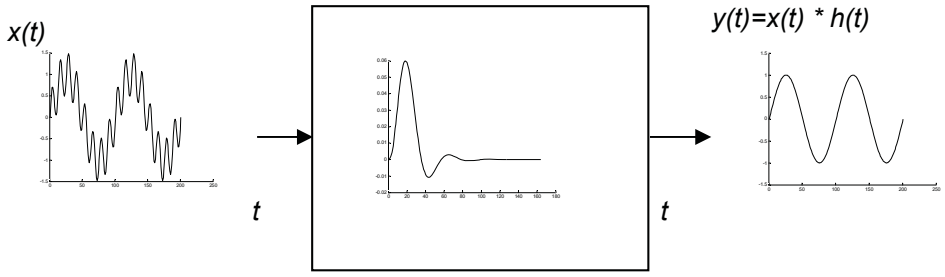
10

Konsep Filter

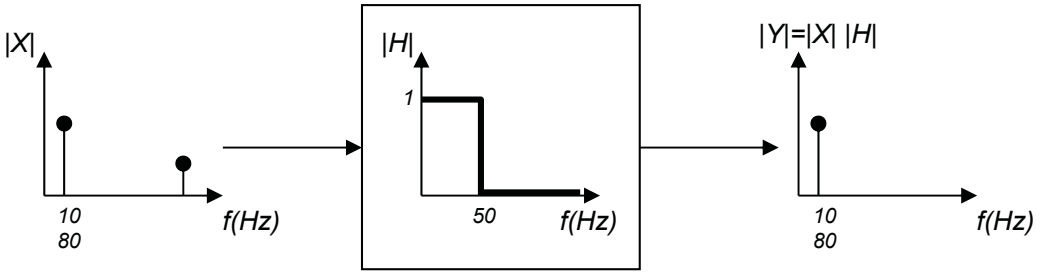
Bab 8 telah memberikan penjelasan tentang sistem diskrit. Sistem diskrit dapat dinyatakan dalam dua bentuk yaitu dalam format *impulse response*, *h(n)*, dan dalam format *transfer function*, *H(z)*. Pada bab ini kita akan belajar cara mendesain suatu sistem diskrit yaitu menemukan nilai dari *h(n)* atau *H(z)*. Karena sistem diskrit yang umum ditemukan dalam pemrosesan sinyal adalah filter digital maka sistem diskrit yang akan didesain adalah filter digital. Bab ini akan memulainya dengan mempelajari konsep dasar dari filter.

Definisi umum dari filter adalah sistem yang secara selektif melakukan modifikasi pada sinyal input baik berupa modifikasi bentuk, amplitudo, dan fase. Filter pada umumnya digunakan untuk meningkatkan kualitas dari sinyal input.

Secara sederhana, filter dapat diartikan sebagai suatu sistem yang berfungsi untuk menyaring sinyal yang masuk. Suatu filter menetukan komponen sinusoidal mana saja dari sinyal input tersebut yang dapat diteruskan ke output.



Gambar 10.1
Filter dalam domain waktu



Gambar 10.2
Filter dalam domain frekuensi

Filter dapat digambarkan dalam domain waktu atau dalam domain frekuensi. Gambar 10.1 menunjukkan sebuah filter yang ditampilkan dalam domain waktu. Input dari filter ini, *x(t)*, memiliki dua komponen sinusoidal. Setelah difilter, maka outputnya hanya memiliki satu komponen sinusoidal. Sinyal output, *y(t)*, merupakan konvolusi dari sinyal input, *x(t)*, dan *impulse response*, *h(t).*

Filter merupakan penyaring komponen sinusoidal sehingga proses filter akan lebih mudah dipahami jika filter digambarkan dalam domain frekuensi. Gambar 10.2 menunjukkan suatu filter $H(j\omega)$ dengan respon frekuensi $|H|$.

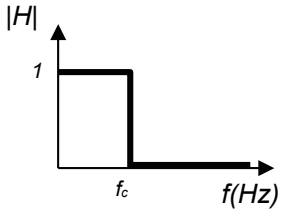
Input pada filter tersebut adalah sinyal $X(j\omega)$ yang memiliki dua komponen sinusoidal pada *10 Hz* dan *80 Hz*. Output, $Y(j\omega)$, merupakan hasil perkalian antara input dan filter, yaitu $|Y| = |X|\ |H|$. Karena respon frekuensi dari filter tersebut bernilai 1 untuk *0 Hz* sampai *50 Hz* dan bernilai nol untuk frekuensi lebih besar dari *50 Hz*, maka filter tersebut hanya melewatkan satu komponen sinusoidal yaitu pada *10 Hz*.

Setelah mempelajari bab ini mahasiswa akan dapat menjelaskan tentang konsep filter, mengidentifikasi jenis, orde dan parameter filter, serta dapat menjelaskan pengaruh pergeseran fase pada suatu proses filter.

10.1 Jenis Filter

Filter dapat diklasifikasi berdasarkan range frekuensi yang disaring atau diteruskan. Secara umum kita mengenal beberapa jenis filter yaitu:

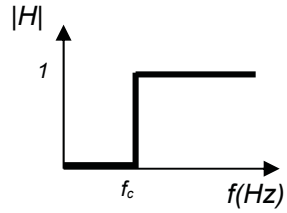
- **Lowpass Filter (LPF)** adalah filter yang hanya meneruskan komponen sinusiodal dari sinyal input yang berfrekuensi rendah ($< f_c$). Filter ini akan menghentikan komponen sinusoidal berfrekuensi tinggi ($> f_c$). Gambar 10.3 menunjukkan respon frekuensi dari LPF.



Gambar 10.3
Respon Frekuensi dari LPF

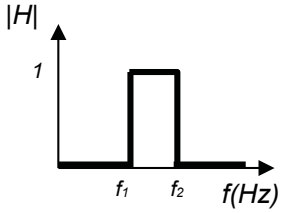
- **Highpass Filter (HPF)** adalah filter yang hanya meneruskan komponen sinusiodal dari sinyal input yang berfrekuensi tinggi ($> f_c$). Filter ini akan menghentikan

komponen sinusoidal berfrekuensi rendah (< f_c). Gambar 10.4 menunjukkan respon frekuensi dari HPF.



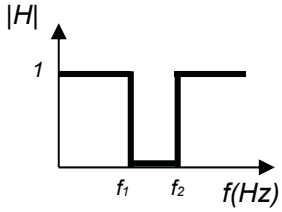
Gambar 10.4
Respon Frekuensi dari HPF

- **Bandpass Filter (BPF)** adalah filter yang hanya meneruskan komponen sinusiodal dari sinyal input yang frekuensinya terletak pada range tertentu ($f_1 < f < f_2$). Filter ini akan menghentikan komponen sinusoidal yang berada di luar range frekuensi tersebut ($f < f_1$ atau $f > f_2$). Gambar 10.5 menunjukkan respon frekuensi dari BPF.



Gambar 10.5
Respon Frekuensi dari BPF

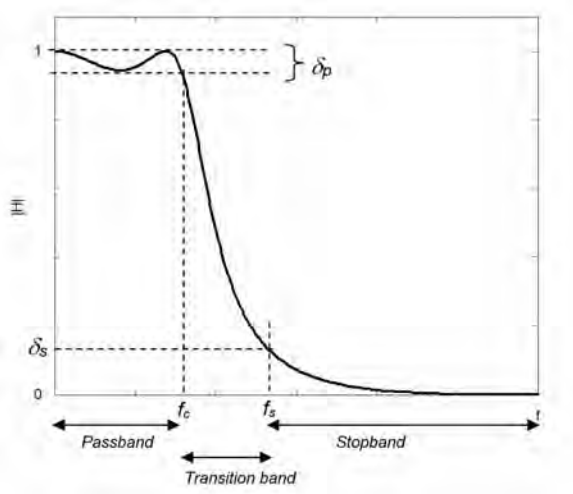
- **Bandstop Filter (BSF)** adalah filter yang hanya tidak meneruskan komponen sinusiodal dari sinyal input yang frekuensinya terletak pada range tertentu ($f_1 < f < f_2$). Filter ini akan meneruskan komponen sinusoidal yang berada di luar range frekuensi tersebut ($f < f_1$ atau $f > f_2$). Gambar 10.6 menunjukkan respon frekuensi dari BSF.



Gambar 10.6
Respon Frekuensi dari BSF

10.2 Parameter Filter

Respon frekuensi dari filter-filter yang ditunjukkan pada Gambar 10.3 sampai 10.6 adalah respon frekuensi yang ideal. Secara praktis, respon frekuensi dari filter memiliki bentuk yang tidak persis sama dengan respon frekuensi ideal. Selain parameter standar dari suatu filter yaitu frekuensi cut-off (f_c atau f_1 dan f_2), kita masih membutuhkan beberapa parameter lain yang menunjukkan kedekatan respon frekuensi dari suatu filter ke respon frekuensi ideal. Gambar 10.7 menunjukkan respon frekuensi dari suatu filter LPF.

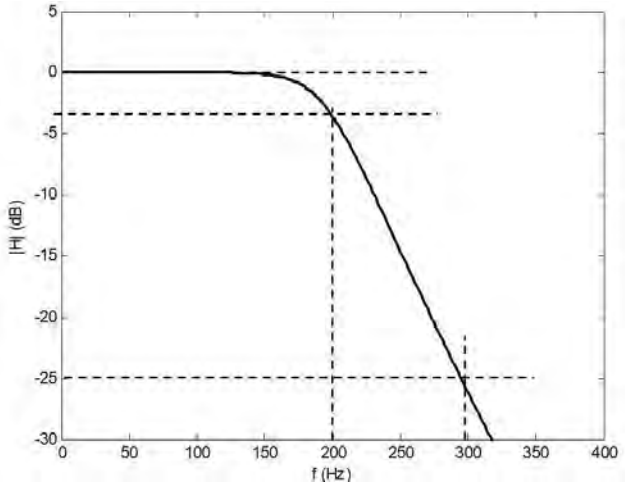


Gambar 10.7
Parameter pada respon frekuensi LPF

Parameter-parameter penting dari suatu LPF:

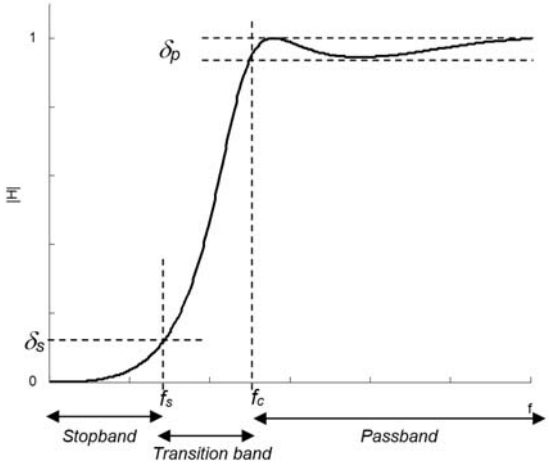
- ***Passband*** adalah daerah frekuensi yang akan diteruskan oleh filter.
- ***Stopband*** adalah daerah frekuensi yang tidak akan diteruskan oleh filter. Komponen sinusoidal pada daerah ini tidak benar-benar disaring oleh filter melainkan telah sangat dilemahkan dengan pelemahan lebih besar atau sama dengan δ_s.
- ***Transition band*** adalah lebar daerah frekuensi maksimumum yang ditoleransi sebagai daerah transisi dari *passband* ke *stopband*. Komponen sinusoidal pada daerah ini masih dilewatkan oleh filter tetapi sudah mengalami sedikit pelemahan.
- ***Frekuensi cut-off (f_c)*** adalah frekuensi yang menentukan batas komponen sinusoidal yang dapat diteruskan oleh filter tersebut. Parameter ini juga dikenal sebagai frekuensi *passband edge* (f_p).
- ***Frekuensi stopband edge (f_s)*** adalah frekuensi yang menentukan batas komponen sinusoidal yang dapat dianggap tidak diteruskan oleh filter tersebut.
- ***Passband ripple (δ_p)*** adalah *ripple* yang masih dapat ditoleransikan pada daerah passband. Passband ripple ini biasanya dinyatakan dalam bentuk desibel (dB).
- ***Stopband attenuation (δ_s)*** adalah pelemahan minimum untuk komponen sinusoidal yang ada di wilayah stopband. *Stopband attenuation* ini biasanya dinyatakan dalam bentuk desibel (dB).

Gambar 10.8 menunjukkan contoh suatu filter LPF dengan $\delta_p = 3\ dB$, $f_c = 200\ Hz$, $f_s = 300\ Hz$, dan $\delta_s = -25\ dB$.



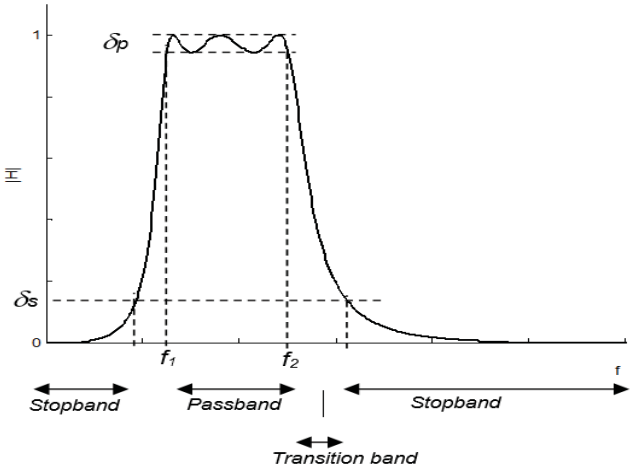
Gambar 10.8
Contoh respon frekuensi LPF

Parameter yang sama juga berlaku pada filter HPF seperti ditunjukkan pada Gambar 10.9.

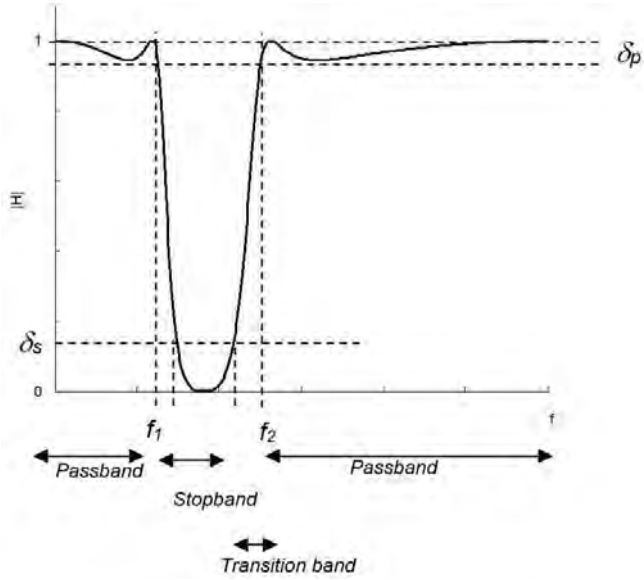


Gambar 10.9
Parameter pada respon frekuensi HPF

Parameter untuk filter BPF dan BSF ditunjukkan pada Gambar 10.10. dan 10.11.



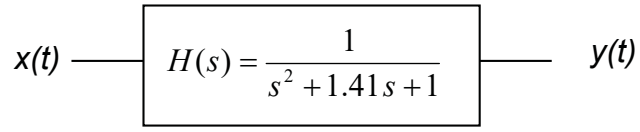
Gambar 10.10
Parameter pada respon frekuensi BPF



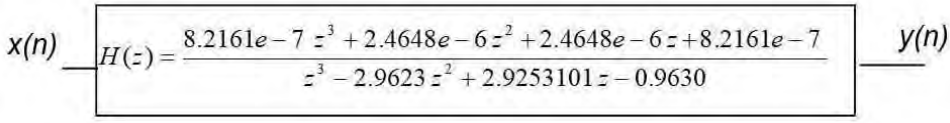
Gambar 10.11
Parameter pada respon frekuensi BSF

10.3 Orde dari Filter

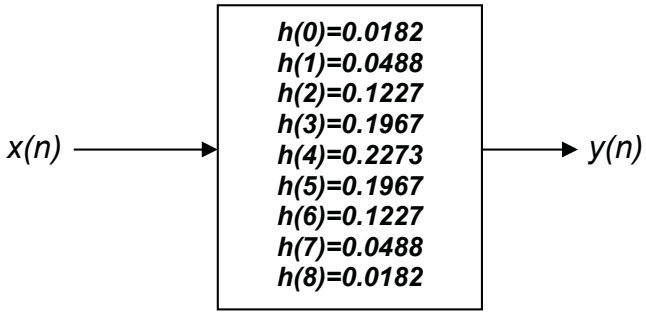
Orde dari filter ditentukan oleh banyaknya koefisien dalam persamaan *transfer function*-nya. Jika suatu filter analog dinyatakan dalam bentuk *H(s)*, maka orde dari filter tersebut adalah pangkat tertinggi dari *s* pada persamaan *H(s)*. Jika suatu filter digital dinyatakan dalam bentuk *transfer function*, *H(z)*, maka orde dari filter itu adalah pangkat tertinggi dari *z*. Suatu filter digital yang dinyatakan dalam bentuk *impulse response* memiliki orde sebesar *N-1*, dimana N adalah panjang dari *h(n)*. Filter pada Gambar 10.12 menunjukan contoh filter dengan berbagai orde.



a)



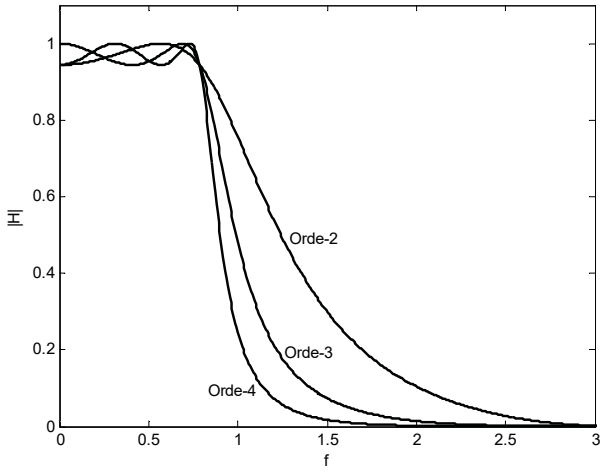
b)



c)

Gambar 10.12
a) Filter analog orde-2, b) Filter digital orde-3, c) Filter digital orde-8

Orde dari filter menentukan kemiripan respon frekuensi suatu filter dengan respon ideal. Semakin tinggi orde suatu filter, semakin sempit pula *transition band*-nya. Filter yang baik memiliki transition band yang sempit seperti ditunjukkan pada Gambar 10.13. Di sisi lain, semakin tinggi orde dari suatu filter, semakin kompleks persamaan *transfer function*-nya, yang akan berakibat pada rumitnya proses implementasi dari filter tersebut.



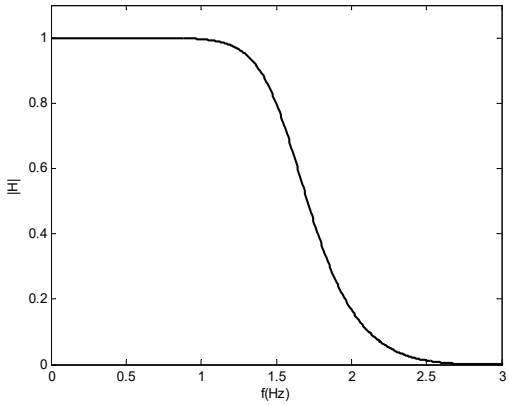
Gambar 10.13
Respon frekuensi filter LPF orde-2, orde-3, dan orde-4

10.4 Filter Analog

Filter dapat diimplementasikan berupa filter analog atau filter digital. Filter analog menggunakan komponen elektronika seperti resistor, kapasitor, dan induktor. Teori tentang filter pertama-tama dikembangkan pada filter analog sehingga untuk dapat memahami konsep lengkap dari filter digital, diperlukan pemahaman karakteristik filter analog. Filter analog dapat dibedakan atas beberapa tipe berdasarkan karakteristik dari parameter-parameternya.

1. Filter Butterworth

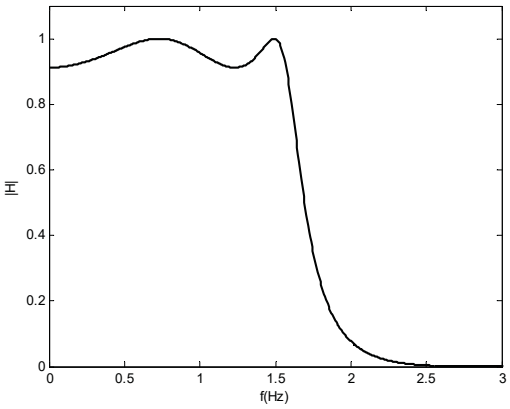
Filter Butterworth diperkenalkan oleh seorang ilmuan Inggris bernama Stephen Butterworth pada 1930. Filter Butterworth memiliki *passband* yang rata (*flat passband*) sehingga filter ini melakukan penguatan yang sama terhadap hampir semua komponen sinusoidal pada daerah *passband*. Kelemahan dari filter ini adalah memiliki kemiringan di daerah transition band yang sangat landai sehingga diperlukan orde yang sangat tinggi untuk mempersempit *transition band*. Gambar 10.14 menunjukkan respon frekuensi dari suatu filter LPF Butterworth.



Gambar 10.14
Filter LPF Butterworth orde-4

2. Filter Chebyshev Tipe I

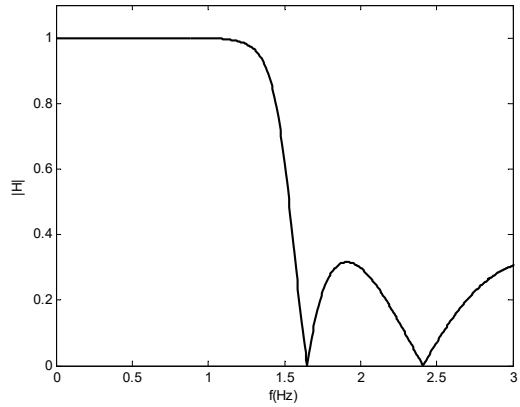
Filter Chebyshev Tipe I diturunkan dari persamaan polinomial yang diperkenalkan oleh Pafnuty Chebyshev. Filter Chebyshev Tipe I ini memiliki *ripple* pada daerah *passband* sehingga filter ini melakukan penguatan yang tidak sama terhadap komponen sinusoidal pada daerah *passband*. Kelebihan filter ini adalah memiliki kemiringan di daerah transition band yang lebih curam sehingga dapat memiliki transition band yang sempit pada orde kecil. Gambar 10.15 menunjukkan respon frekuensi dari suatu filter LPF Chebyshev Tipe I.



Gambar 10.15
Filter LPF Chebyshev Tipe I orde-4

3. <u>Filter Chebyshev Tipe II</u>

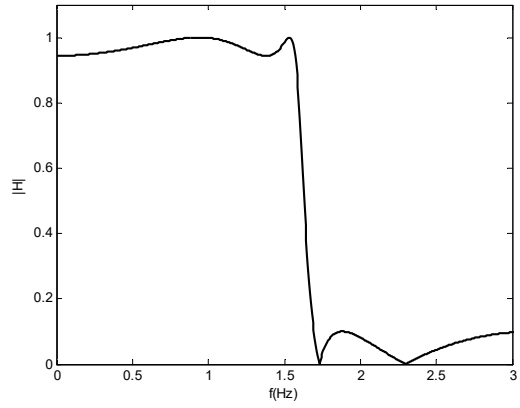
Filter Chebyshev Tipe II diturunkan dari persamaan inverse polinomial Chebyshev. Filter Chebyshev Tipe II bertujuan untuk membuat filter dengan karakteristik transition band yang sama curam dengan Chebyshev Tipe I, tetapi memiliki passband yang rata. Sebagai konsekuensinya, filter ini memiliki *ripple* pada daerah stopband sehingga pelemahan pada daerah stopband δ_s tidak cukup baik untuk memfilter komponen sinusoidal yang tidak ingin dilewatkan. Gambar 10.16 menunjukkan respon frekuensi dari suatu filter LPF Chebyshev Tipe II.



Gambar 10.16
Filter LPF Chebyshev Tipe II orde-4

4. <u>Filter Elliptic</u>

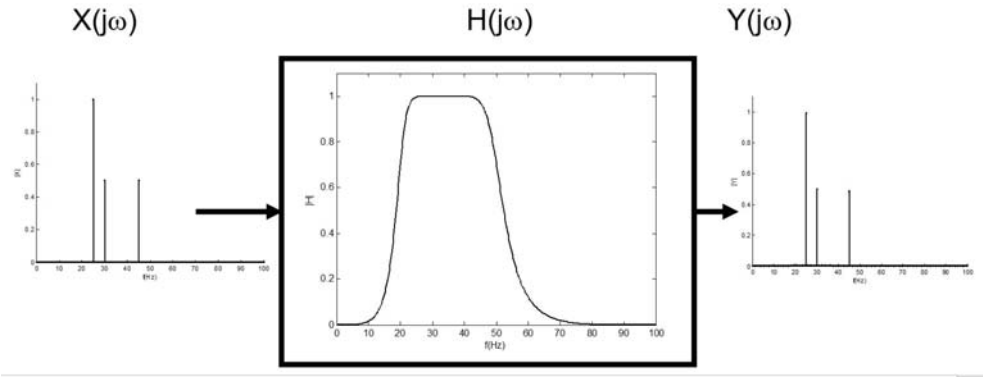
Filter Elliptic juga dikenal dengan nama filter Cauer merupakan filter yang dengan transition band yang curam tetapi memiliki ripple pada passband dan stopband. Besarnya ripple ini ditentukan pada saat filter ini dirancang. Jika ripple pada passband dibuat kecil maka ripple pada stopband akan besar, demikian sebaliknya. Gambar 10.17 menunjukkan respon frekuensi dari suatu filter LPF Elliptic.



Gambar 10.17
Filter LPF Elliptic orde-4

10.5 Pergeseran Fase pada Filter

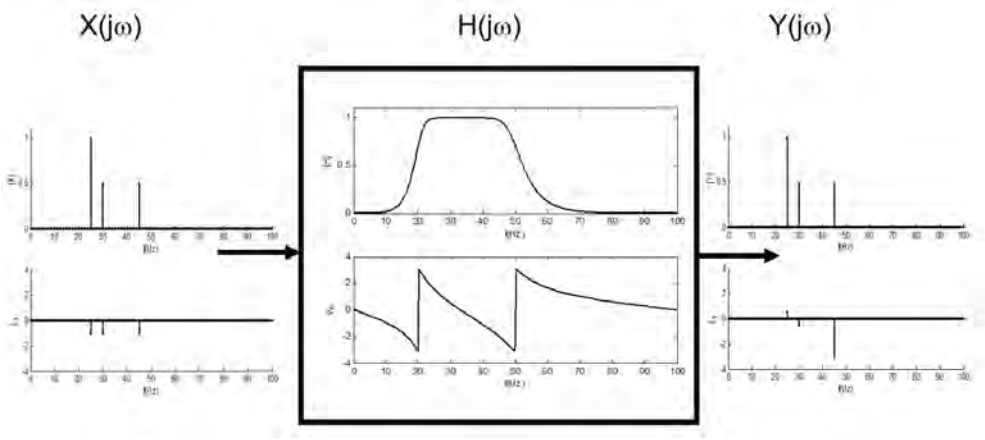
Sejauh ini kita telah mempelajari bagaimana filter dapat meneruskan atau menyaring komponen-komponen sinusoidal dari sinyal input. Hal lain yang perlu mendapat perhatian dari kerja filter yaitu efek dari pergeseran fase pada komponen-komponen sinusoidal.



Gambar 10.18
Proses filter pada sinyal input dengan dua komponen sinusoidal

Gambar 10.18 menunjukkan suatu filter BPF dengan *passband* dari 20 sampai 50 Hz. Karena sinyal input hanya mempunyai tiga komponen sinusoidal dengan frekuensi yang terletak dalam daerah *passband*, maka filter ini meneruskan semua komponen sinusoidal dari sinyal input, dengan kata lain $|Y| = |X|$.

Kita tentunya berharap bahwa sinyal output, *y(t),* akan sama dengan sinyal input, *x(t)*, tetapi ternyata kita bisa mendapati $y(t) \neq x(t)$. Penyebabnya adalah karena fase sinyal output tidak sama dengan fase sinyal input, $\varphi_y \neq \varphi_x$. Dengan kata lain, telah terjadi pergeseran fase pada ouput.

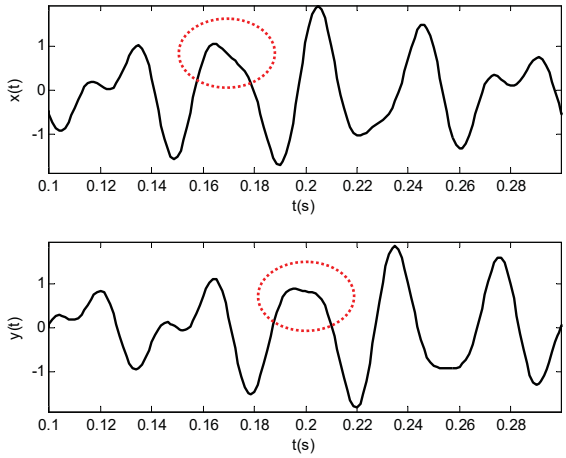


Gambar 10.19
Pergeseran fase pada proses filter

Gambar 10.19 menunjukkan bahwa ketiga komponen sinusoidal pada sinyal input, *x(t)*, sama-sama memiliki fase sebesar *-1 rad.* Setelah dilewatkan pada filter, amplitudo kedua komponen ini tetap sama tetapi fasenya telah berubah menjadi *0.5 rad* untuk komponen pertama, *-0.5 rad* untuk komponen kedua, dan *-3 rad* untuk komponen ketiga. Pergeseran ini terjadi karena fase output merupakan penjumlahan fase input dan fase dari filter.

$$\varphi_y = \varphi_x + \varphi_h$$

Pergeseran ini akan menyebabkan $y(t) \neq x(t)$ seperti ditunjukkan pada Gambar 10.20, khususnya pada daerah yang dilingkari.



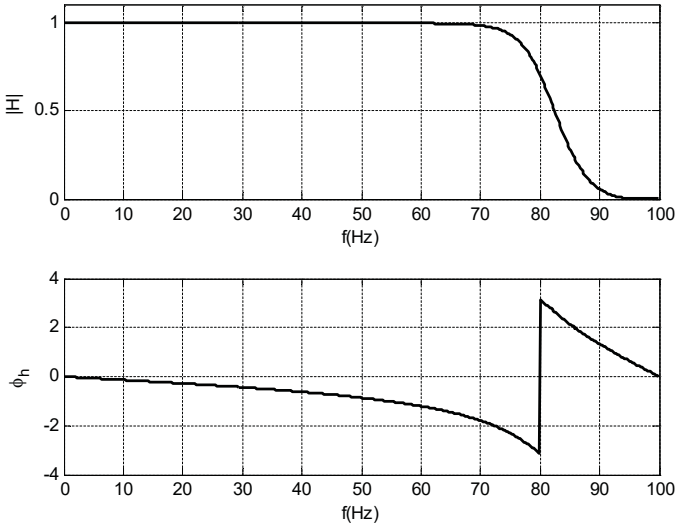
Gambar 10.20
Perbedaan bentuk input dan output karena pergeseran fase

Jika suatu filter digunakan pada aplikasi yang mementingkan bentuk sinyal maka pergeseran fasa ini dapat merusak bentuk sinyal sekalipun pada daerah *passband*. Kita harus berhati-hati dalam menggunakan sebuah filter untuk menghilangkan noise pada sinyal jantung (ECG). Jika pergeseran fase ini tidak kita atur dengan baik maka filter tersebut bukan hanya menghilangkan noise tetapi juga merusak bentuk sinyal ECG.

10.6 Group Delay

Masalah pergeseran fase di atas dapat diatasi jika kita menggunakan filter yang sama sekali tidak mengubah fase dari komponen sinusoidal sinyal input, atau dengan kata lain, respon fase dari filter, φ_h, selalu nol di daerah *passband*. Harapan ini tidak mungkin diimplementasikan karena semua filter melakukan perubahan fase pada sinyal input.

Karena pergeseran fase dari suatu komponen sinusoidal menentukan *delay* yang terjadi pada komponen sinusoidal itu, maka pergeseran fase yang terjadi pada suatu filter tidak akan merusak bentuk sinyal input jika semua komponen sinusoidal di daerah passband di-*delay* dengan waktu yang sama. Kerusakan bentuk sinyal terjadi jika komponen-komponen sinusoidal di-*delay* dengan waktu yang berbeda-beda. Semua komponen sinusiodal akan di-*delay* dengan waktu yang sama jika respon fase dari suatu filter berbentuk garis lurus (linier).



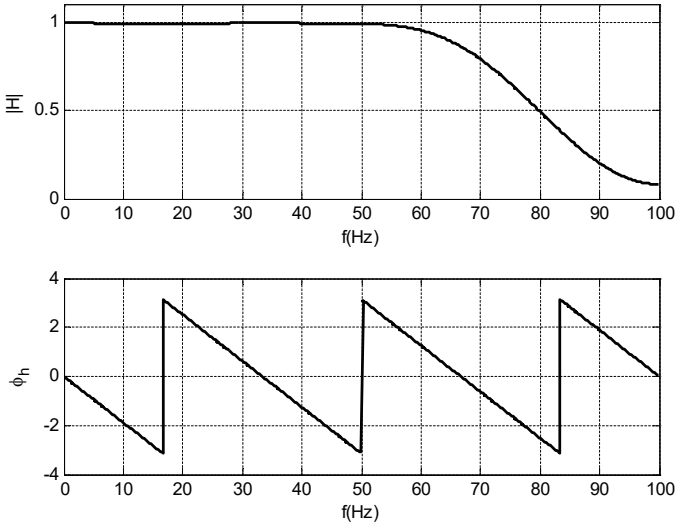
Gambar 10.21
Filter dengan pergeseran fase tidak linier

Gambar 10.21 menampilkan respon dari suatu filter dengan respon fase yang tidak linier (tidak berbentuk garis lurus). Tabel 10.1 menampilkan pergeseran fase dan *delay* dari beberapa komponen sinusoidal pada filter tersebut. Tabel tersebut menunjukkan bahwa komponen sinusoidal di-*delay* dengan waktu yang berbeda-beda.

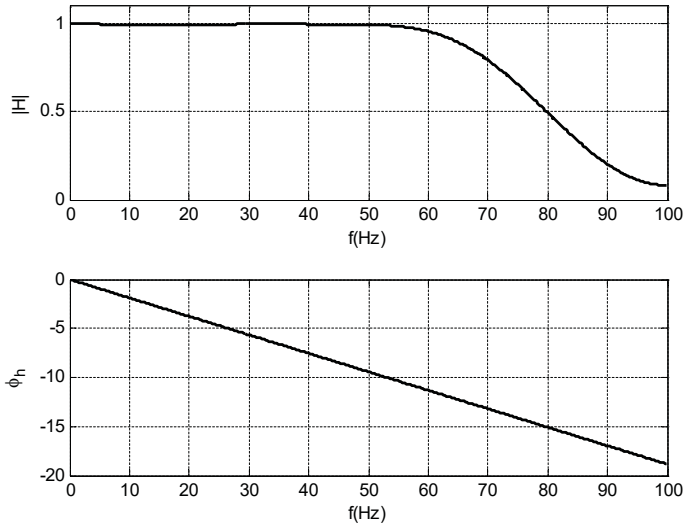
Tabel 10.1
Pergeseran fase dan delay dari filter Gambar 10.20

Komponen sinusoidal (Hz)	Pergeseran fase (radian)	Delay (ms)
20	-0.276	2.2
30	-0.434	2.3
40	-0.622	2.5
50	-0.862	2.7
60	-1.206	3.2
70	-1.792	4.1

Gambar 10.22 menampilkan respon dari suatu filter dengan respon fase yang linier (berbentuk garis lurus). Respon fase dari filter tersebut terlihat seperti gelombang gergaji karena hanya digambarkan dari -π sampai π. Sesungguhnya kurva respon fase tersebut berbentuk garis lurus seperti pada Gambar 10.23. Tabel 10.2 menampilkan pergeseran fase dan *delay* dari beberapa komponen sinusoidal pada filter tersebut. Tabel tersebut menunjukkan bahwa semua komponen sinusoidal di-*delay* dengan waktu yang sama.



Gambar 10.22
Filter dengan pergeseran fase linier



Gambar 10.23
Filter dengan pergeseran fase linier yang digambar dalam bentuk garis lurus

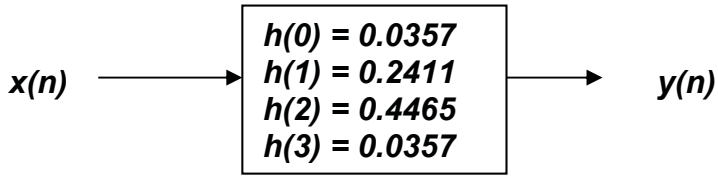
Tabel 10.2
Pergeseran fase dan delay dari filter Gambar 10.22

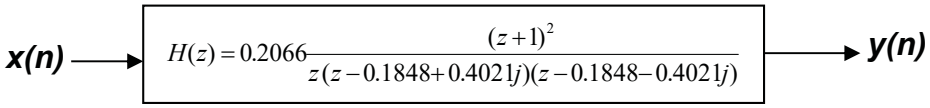
Komponen sinusoidal (Hz)	Pergeseran fase (radian)	Delay (ms)
20	-3.770	30
30	-5.655	30
40	-7.540	30
50	-9.4248	30
60	-11.3104	30
70	-13.1944	30

Jika respon fase dari suatu filter bersifat linier maka semua komponen sinusoidal dalam sinyal input akan di-*delay* dengan waktu yang sama sehingga tidak akan merusak bentuk sinyal. Besarnya waktu *delay* yang terjadi karena perseseran fase ini disebut ***group delay***. Filter pada tabel 10.2 memiliki *group delay* sebesar 30 ms.

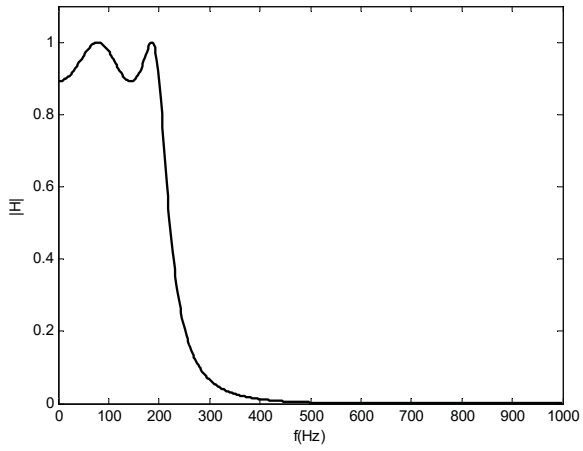
SOAL LATIHAN

1. Tentukan orde dari filter-filter di bawah ini.

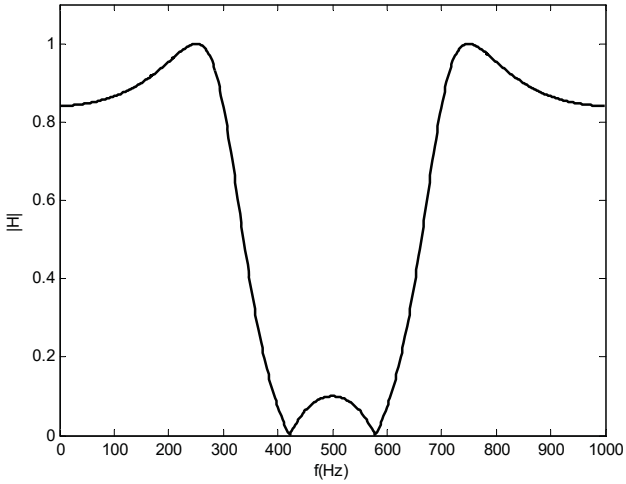




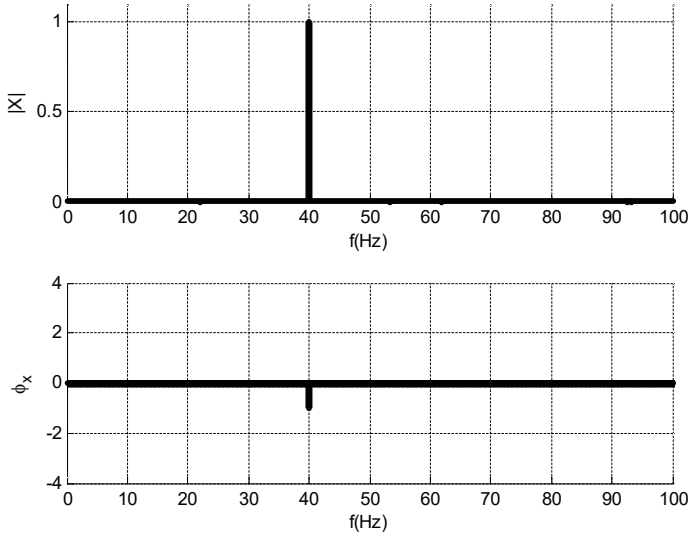
2. Jika filter berikut ini memiliki δ_s = *-14 dB*, tentukan parameter dari filter tersebut (dalam dB dan radian). Tentukan juga jenis dari filter ini.

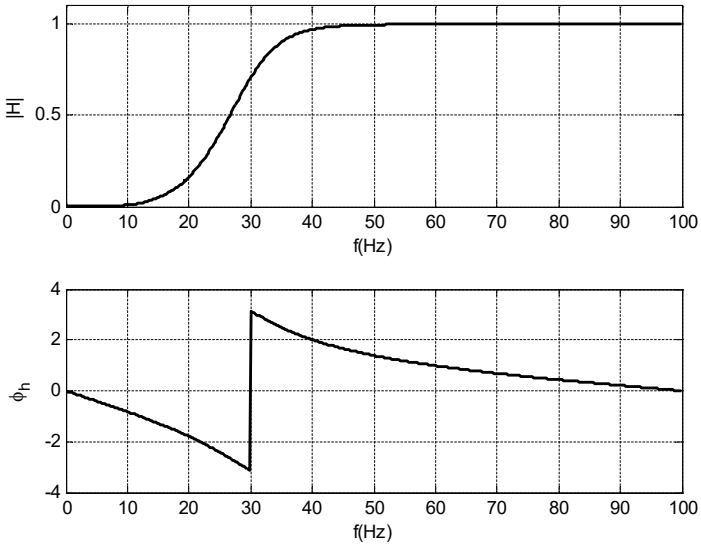


3. Tentukan jenis dan parameter dari filter berikut ini (dalam dB dan radian).



4. Jika sinyal *x(t)* berikut ini diinputkan pada filter dengan respon frekuensi dan fasa seperti gambar-gambar di bawah ini, gambarlah sinyal output dalam domain waktu, *y(t)*, dan dalam domain frekuensi $|Y|$ dan φ_y.

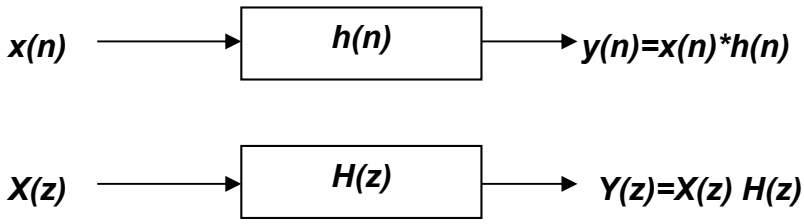



|H|
1
0.5
0
0 10 20 30 40 50 60 70 80 90 100
f(Hz)
ϕ_h
4
2
0
-2
-4
0 10 20 30 40 50 60 70 80 90 100
f(Hz)

11

Filter Digital

Setelah kita mempelajari konsep filter dan filter analog pada Bab 9 dan 10 maka sekarang kita akan mempelari tentang konsep filter digital. Setelah mempelajari bab ini maka mahasiswa akan dapat mengidentifikasi kelebihan dan kekurangan dari filter digital jika dibandingkan dengan filter analog, menentukan parameter dari filter digital, menggambarkan respon frekuensi dari filter digital, dan dapat merancang implementasi filter digital dalam bentuk perangkant keras.



Gambar 11.1
Filter digital dalam bentuk impulse respons dan transfer function

Filter digital adalah sistem diskrit (Bab 8) yang berfungsi sebagai filter. Karena merupakan sistem diskrit maka filter

digital dapat dinyatakan dalam bentuk *impulse response*, *h(n)*, atau *transfer function*, *H(z)*, seperti pada Gambar 11.1. Filter digital juga memiliki jenis yang sama dengan filter analog yaitu LPF, HPF, BPF dan BSF.

11.1 Kelebihan Filter Digital

Filter analog berupa persamaan yang diimplementasikan dalam bentuk rangkaian elektronika. Filter digital berisi angka-angka yang diproses secara aritmetika (penjumlahan, pengurangan, perkalian, dan pembagian). Filter digital ini diimplementasikan dalam bentuk program komputer atau proses digital di komputer atau mikroprosesor. Karena perbedaan ini maka filter digital memiliki beberapa kelebihan dibandingkan dengan filter analog antara lain:

- Karena hanya berupa angka dan proses aritmetika maka filter digital *mudah diimplementasikan* sekalipun dengan orde yang sangat tinggi. Penambahan satu orde pada filter analog berakibat pada penambahan beberapa komponen elektronika yang akan membuat fisik rangkaian menjadi lebih besar. Penambahan orde pada filter digital hanya akan menambah beberapa angka pada program komputer atau prosesor. Kita akan jarang menjumpai filter analog dengan orde lebih dari 6, sedangkan kita akan sering menjumpai filter digital ber-orde puluhan bahkan ratusan.

- Filter digital bisa *memiliki respon frekuensi yang sangat bervariasi*. Filter analog terbatas pada bentuk respon frekuensi seperti Butterworth, Chebyshev, dan Elliptic karena batasan ketersediaan jenis dan nilai komponen elektronika. Karena hanya berupa angka maka respon frekuensinya dapat divariasi tanpa batasan fisik komponen. Kita dapat mendesain filter digital dengan fase yang benar-benar linier. Hal ini tidak dimungkinkan oleh filter analog.

- Filter digital *tidak dipengaruhi oleh suhu* atau gangguan luar lainnya. Nilai komponen dari filter analog dapat berubah jika suhu berubah. Hal ini tidak terjadi pada filter digital karena angka-angkanya ditentukan oleh progam yang dituliskan pada prosesor.

- *Parameter dari filter digital bisa diubah dengan gampang* karena hanya berupa angka-angka pada program komputer atau mikroprosesor. Perubahan parameter pada filter analog harus dilakukan dengan kengganti komponen elektronikanya. Kelebihan ini juga memungkinkan implementasi dari adaptive filter.

- Filter digital *dapat disimpan (save) dan diduplikasi* dengan tidak ada perubahan performansi. Hal ini tidak dijamin oleh filter analog karena nilai dari komponen elektronika selalu bervariasi.

- Filter digital *dapat digunakan pada frekuensi yang sangat rendah*. Hal ini sulit dilakukan oleh filter analog karena membutuhkan nilai kapasitor yang sangat besar.

11.2 Kelemahan Filter Digital

Disamping kelebihan-kelebihan di atas, filter digital juga memiliki beberapa kelemahan jika dibandingkan dengan filter analog, antara lain:

- Filter digital *memiliki range frekuensi yang terbatas* dan cenderung lebih rendah dari filter analog. Range frekuensi filter digital hanya dibatasi sampai setengah dari frekuensi samplingnya. Frekuensi sampling sangat ditentukan oleh kecepatan dari ADC dan mikroprosesor yang digunakan.

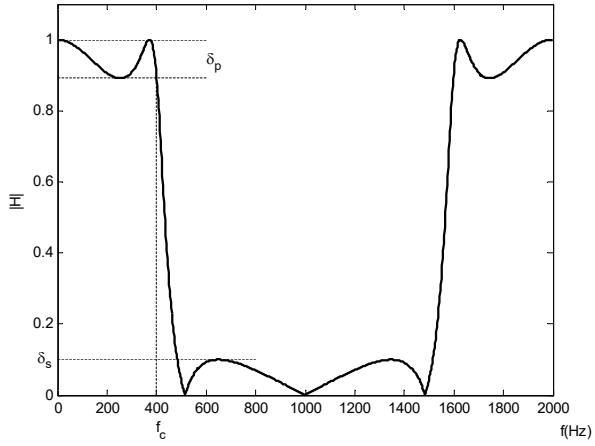
- Filter digital memiliki *noise yang disebabkan oleh pembulatan angka-angka* di dalam prosesnya. Semakin

tinggi orde dari suatu filter digital maka semakin besar pula efek dari pembulatan khususnya bila ada proses recursive.

- Filter digital tidak dapat digunakan sebagai filter untuk anti aliasing karena filter ini harus diletakkan di daerah analog.
- Filter digital tidak bisa digunakan untuk memfilter sinyal dengan daya tinggi misalnya filter tegangan jala-jala.

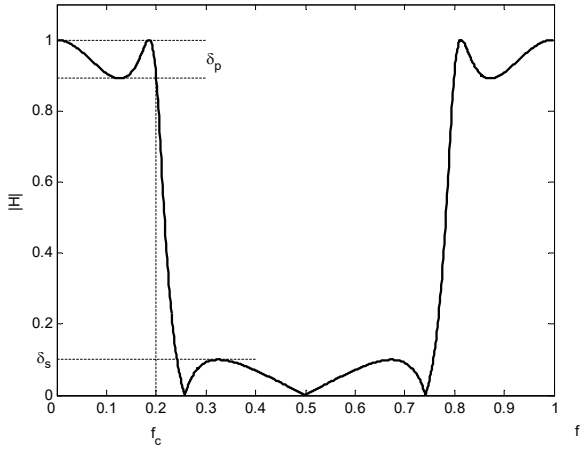
11.3 Parameter Filter Digital

Filter digital memiliki parameter yang sama dengan filter analog tetapi mengalami pencerminan pada f_s */2* seperti dijelaskan pada Bab 4. Gambar 11.2 adalah filter digital LPF dengan f_c = *400 Hz* dan frekuensi sampling, f_s, sebesar *2 kHz*. Jika kita tidak menyadari adanya pencerminan respon frekuensi pada spektrum diskrit maka kita bisa keliru menganggap bahwa gambar ini adalah respon frekuensi dari BSF.

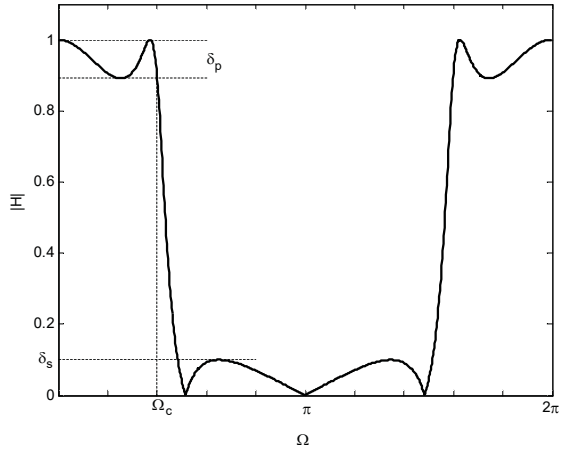


Gambar 11.2
Respon frekuensi filter digital LPF

Respon frekuensi dari filter digital sering digambarkan dalam frekuensi yang sudah dinormalisasi dimana frekuensi sampling bernilai *1* seperti pada Gambar 11.3, atau dalam skala radian ternormalisasi dimana frekuensi sampling adalah 2π dan setengah frekuensi sampling menjadi π seperti pada Gambar 11.4.

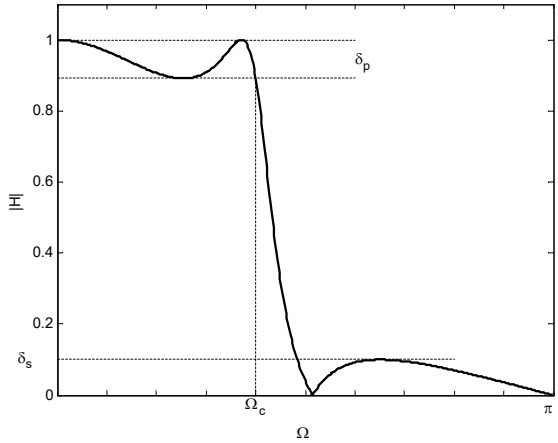


Gambar 11.3
Respon frekuensi filter digital LPF dengan frekuensi ternormalisasi



Gambar 11.4
Respon frekuensi filter digital LPF dengan frekuensi radian ternormalisasi

Karena respon frekuensi pada Gambar 11.4 bersifat pencerminan pada $\Omega = \pi$, maka pada umumnya hanya digambarkan sampai dengan π seperti pada Gambar 11.5.



Gambar 11.5
Respon frekuensi filter digital LPF dengan $\Omega_c = 0.4\ \pi$ (sama dengan $0.2\ f_s$)

Catatan penting untuk pengguna Matlab:
Desain filter digital pada Matlab menggunakan input berupa frekuensi yang sudah dinormalisasi (seperti Gambar 11.3), tetapi frekuensi 1 bukan mewakili f_s melainkan mewakili ½ f_s.

Jika kita menggunakan Matlab untuk mendesain LPF dengan $f_c = 400\ Hz$ pada $f_s = 2000\ Hz$, maka f_c ternormalisasi adalah 0.4 (bukan 0.2).

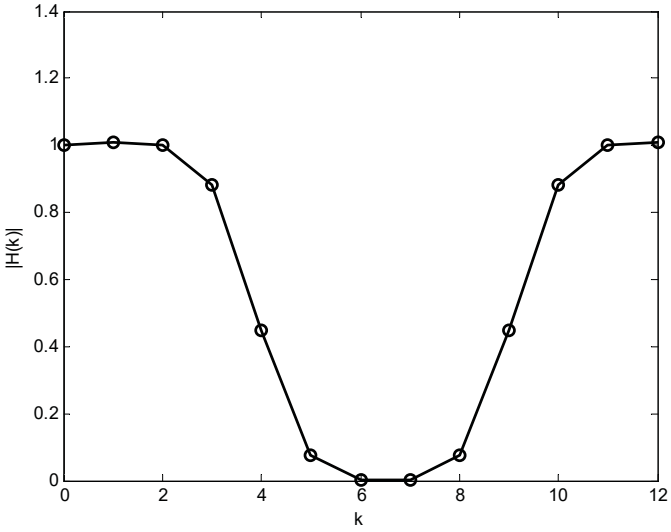
11.4 Menemukan Respon Frekuensi dari Filter

Gambar-gambar di atas adalah kurva respon frekuensi dari filter digital. Bagaimanakah kita bisa menemukan kurva respon frekensi dari suatu filter? Jika kita mengetahui *impulse respons*-nya, *h(n)*, maka respon frekuensinya, *H(k)* dapat dihitung dengan menggunakan DFT seperti pada Bab 4. Jika kita mengetahui transfer functionnya, *H(z)*, maka terlebih dahulu kita harus menemukan *h(n)* dari filter tersebut kemudian baru menghitung respon frekuensinya, *H(k)*, dengan menggunakan DFT seperti pada Bab 4. Cara menemukan *h(n)* sudah dijelaskan di Bab 9. Sebagai contoh kita akan menemukan kurva respon frekuensi dari filter dengan *h(n)* seperti pada Tabel 11.1

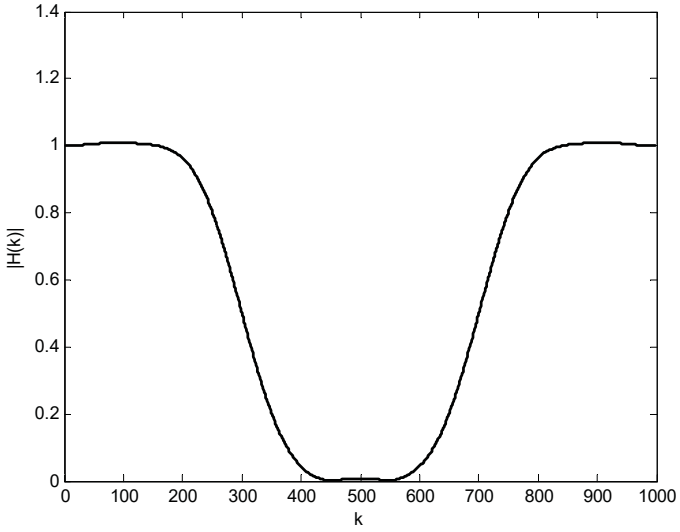
Tabel 11.1
Impulse response dari sebuah filter

n	*h(n)*
0	-0.0041
1	0.0000
2	0.0236
3	-0.0338
4	-0.0724
5	0.2853
6	0.6027
7	0.2853
8	-0.0724
9	-0.0338
10	0.0236
11	0.0000
12	-0.0041

Perhitungan DFT menghasilkan |*H(k)*| sebanyak 12 titik seperti digambarkan pada Gambar 11.6a. Gambar respon frekuensi ini dapat dibuat lebih baik dengan *zero padding*. Gambar 11.6b menunjukkan respon frekuensi yang dihitung menggunakan DFT dengan *zero padding*.



a)

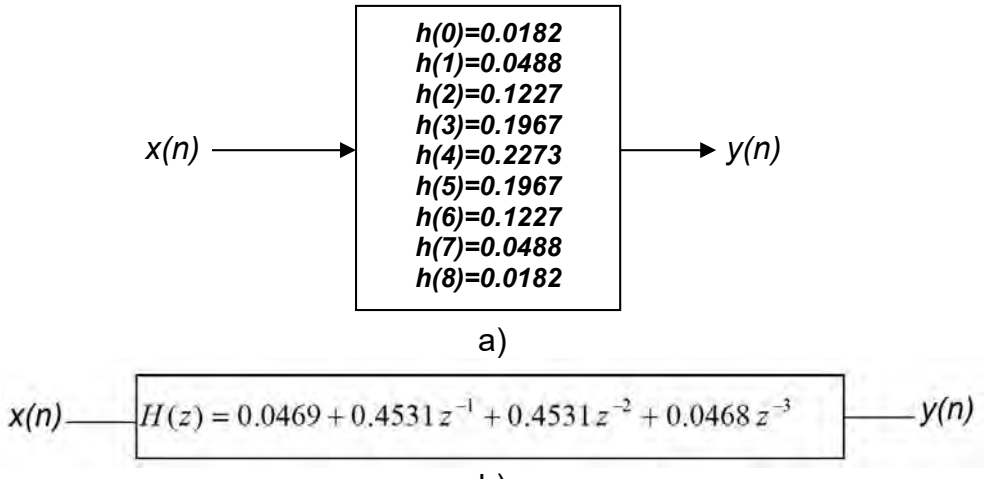


b)

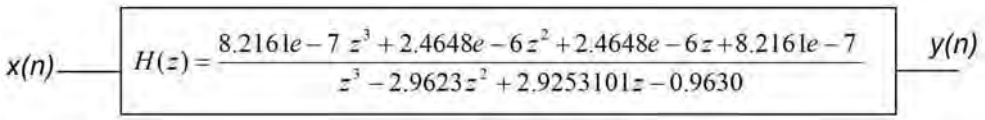
Gambar 11.6
Respon frekuensi filter pada Tabel 11.1 a) dihitung dengan DFT; b) dihitung dengan DFT dan zero padding

11.5 Tipe dari Filter Digital

Secara umum, filter digital dibedakan atas dua tipe yaitu *Finite Impulse Response* (FIR) dan *Infinite Impulse Response* (IIR). Sesuai dengan namanya, filter FIR memiliki *impulse response*, *h(n)*, dalam jumlah yang tertentu dan terbatas, sedangkan filter IIR memiliki *h(n)* yang sangat banyak. Filter FIR bisa dinyatakan dalam bentuk *impulse response* atau dalam bentuk *transfer function*, sedangkan filter IIR hanya dinyatakan dalam bentuk *transfer function* karena memiliki *h(n)* yang sangat banyak. Gambar 11.7 menunjukkan contoh filter FIR sedangkan Gambar 11.8 menunjukkan contoh filter IIR.



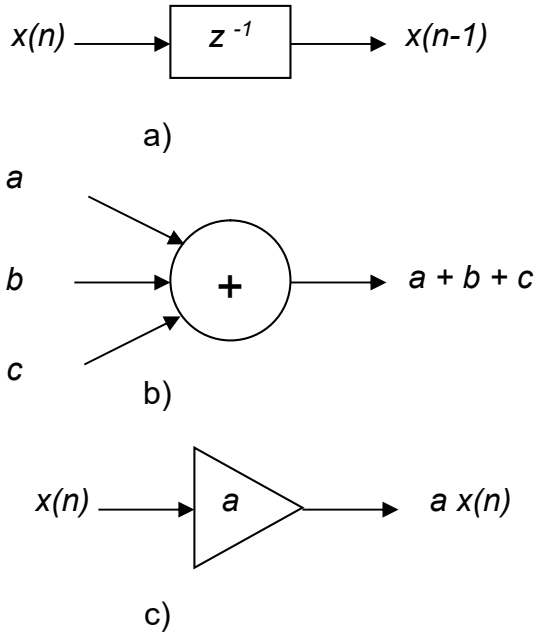
Gambar 11.7
Contoh filter FIR a) dengan impulse respons, b) dengan transfer function



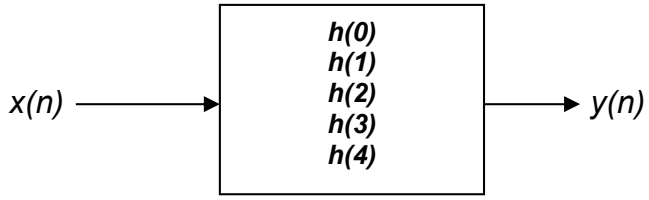
Gambar 11.8
Contoh filter IIR

11.6 Implementasi Filter FIR

Baik filter FIR maupun filter IIR dapat diimplementasikan dalam bentuk program komputer atau perangkat keras digital. Implementasi dalam bentuk program komputer sama dengan implementasi sistem diskrit (Bab 8). Implementasi dalam bentuk perangkat keras menggunakan tiga komponen utama yaitu *delay*, penjumlah, dan pengali. Komponen *delay* menunda input sebesar 1 hitungan. Komponen penjumlah menjumlahkan semua input, sedangkan komponen pengali mengeluarkan output yang merupakan hasil perkalian antara input dengan faktor pengalinya, seperti ditunjukkan pada Gambar 11.9.



Gambar 11.9
Komponen dasar filter digital

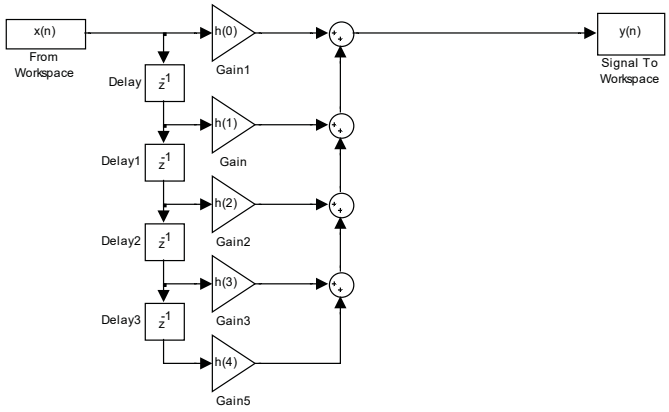


Gambar 11.10
Contoh Filter FIR orde 4

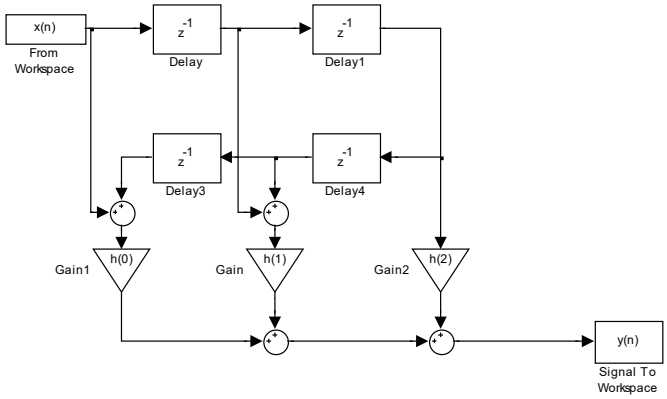
Sebagai contoh, sebuah filter FIR seperti pada Gambar 11.10, dapat dinyatakan dalam bentuk impulse response sehingga outputnya dihitung dengan konvolusi (lihat Bab 6) yaitu:

$$y(n) = h(0)\ x(n) + h(1)\ x(n-1) + h(2)\ x(n-2) + h(3)\ x(n-3) + \ldots + h(N)\ x(n-N)$$

Filter tersebut dapat diimplementasikan dengan struktur transversal (Gambar 11.11) atau struktur fase linier (Gambar 11.12). Struktur fase linier lebih menghemat proses perhitungan karena filter FIR dengan fase linier memiliki *h(0)* yang selalu sama dengan *h(N), h(1) = h(N-1)* dan seterusnya seperti terlihat pada filter di Gambar 11.7a.



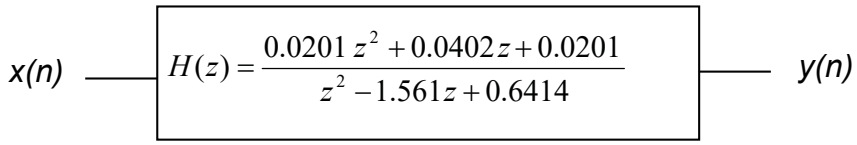
Gambar 11.11
Implementasi FIR dengan struktur transversal



Gambar 11.12
Implementasi FIR dengan struktur fase linier

11.7 Implementasi Filter IIR

Filter IIR juga diimplementasikan dengan menggunakan komponen yang sama seperti pada Gambar 11.9. Filter IIR dapat diimplementasikan dengan beberapa struktur. Struktur yang paling umum dipakai adalah Direct Form I dan Direct Form II.

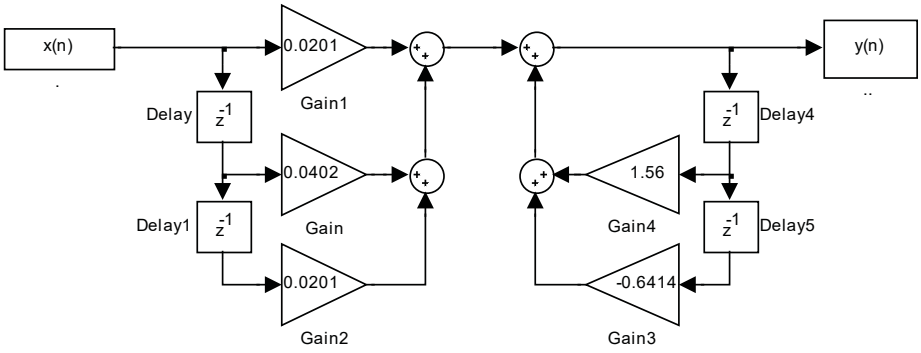


Gambar 11.13
Contoh Filter IIR orde 2

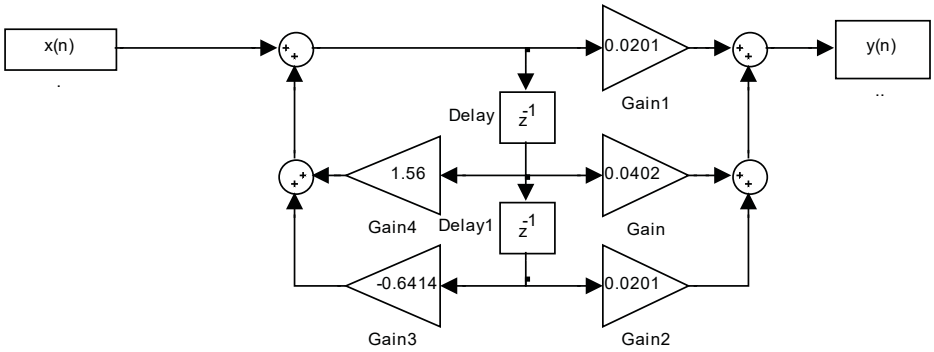
Output dari filter IIR seperti pada Gambar 11.13 dapat dihitung dengan persamaan diferensial:

$$y(n) = 0.0201x(n) + 0.0402x(n-1) + 0.0201x(n-2) + 1.56\,y(n-1) - 0.6414\,y(n-2)$$

Implementasi filter tersebut dengan struktur Direct Form I ditunjukkan pada Gambar 11.14, sedangkan implementasi dengan struktur Direct Form II ditunjukan pada Gambar 11.15. Direct Form II lebih menghemat penggunaan proses *delay*.



Gambar 11.14
Implementasi IIR dengan struktur Direct Form I



Gambar 11.15
Implementasi IIR dengan struktur Direct Form II

11.8 Perbandingan antara FIR dan IIR

Ketika kita merancang filter digital, kita harus memilih tipe apa yang akan digunakan, FIR atau IIR. Kedua tipe ini memiliki kelebihan dan kekurangan. Berikut ini adalah perbandingan antara kedua filter tersebut.

- Filter FIR dapat memiliki respon fase yang benar-benar linier sehingga tidak merusak bentuk sinyal seperti dijelaskan pada Bab 10. Filter IIR tidak memiliki respon fase yang linier. Filter FIR digunakan untuk aplikasi yang sangat mementingkan bentuk sinyal misalnya pada transmisi data digital, biosignal (ECG, EEG, dsb), audio digital, dan pemrosesan citra digital. Jika kita mentransmisi data digital (gelombang kotak) melewati sebuah filter IIR maka outputnya tidak lagi berbentuk kotak. Distorsi ini tidak terjadi pada filter FIR karena memiliki respon fase yang linier.

- Filter FIR selalu stabil karena output hanya tergantung dari input, sedangkan output dari filter IIR tergantung dari input dan output sebelumnya sehingga bisa tidak stabil.

- IIR menggunakan proses umpanbalik (recursive) sehingga rawan terhadap pembulatan angka pada proses perhitungan output. Filter FIR tidak terlalu terpengaruh oleh pembulatan angka.

- Untuk mendapat *transition band* yang curam diperlukan filter FIR dengan orde yang sangat tinggi. Filter IIR tidak memerlukan orde yang tinggi untuk mendapat transition band yang curam.

- Filter IIR bisa mempunyai respon frekuensi yang persis sama dengan filter analog. Jika kita ingin mengganti filter analog pada suatu sistem dengan filter digital tanpa

merubah respon frekuensinya maka kita harus memiliki filter IIR.

- Proses desain filter FIR sangat memerlukan bantuan komputer karena biasanya filter FIR ber-orde sangat tinggi sehingga sangat banyak angka yang harus dihitung. Filter IIR dapat didesain secara manual tanpa bantuan komputer.

- Karena memiliki orde yang sangat tinggi maka filter FIR mengalami *delay* yang sangat panjang dibandingkan dengan filter IIR

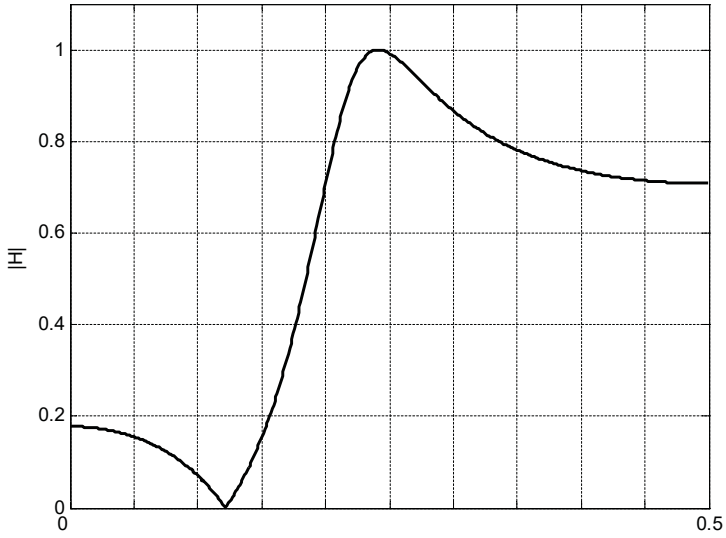
Gunakan filter IIR jika filter kita hanya mementingkan kecuraman transition band. Gunakan FIR jika kita harus menghindari distorsi fase dan jika kita memiliki komputer atau prosesor yang dapat menghitung filter dengan orde sangat tinggi.

SOAL LATIHAN

1. Gambarlah respon frekuensi dari suatu filter HPF digital dengan $\Omega_c = 0.1\pi$, $\delta_p = 2$ dB, dan $\delta_s = -10$ dB.

2. Gambarlah respon frekuensi dari suatu filter LPF digital (dalam bentuk frekuensi radian ternormalisasi) yang memiliki $f_c = 1$ kHz, $fs = 5$ kHz, $\delta_p = 3$ dB, dan $\delta_s = -20$ dB.

3. Respon frekuensi dari suatu filter digital digambarkan dalam bentuk frekuensi ternormalisasi seperti di bawah ini. Tentukan jenis dan parameter filter tersebut dalam Hz dan dB.



4. Implementasilkan filter dengan *h(n)* berikut ini pada Simulink (dengan modifikasi file `soal10_4.mdl`) menggunakan struktur transversal. Gunakan filter tersebut untuk input *s* yang terdapat dalam file `sinyal10_4.mat`. Amati bentuk input dan outputnya tentukan jenis dan parameter dari filter tersebut.

n	*h(n)*
0	0.0068
1	0.0152
2	0.0395
3	0.0780
4	0.1212
5	0.1553
6	0.1682
7	0.1553
8	0.1212
9	0.0780
10	0.0395
11	0.0152
12	0.0068

5. Ulangi soal nomor 4 di atas dengan menggunakan struktur fase linier.

6. Implementasilkan filter dengan *H(z)* berikut ini pada Simulink dengan menggunakan struktur Direct Form I. Gunakan filter tersebut untuk input x yang terdapat dalam file `sinyal10_4.mat`. Amati bentuk input dan outputnya tentukan jenis dan parameter dari filter tersebut.

$$H(z) = \frac{0.0191\,z^3 - 0.0185\,z^2 - 0.0185z + 0.0191}{z^3 - 2.8665\,z^2 + 2.7534\,z - 0.8856}$$

7. Ulangi soal nomor 6 dengan mengunakan struktur Direct Form II.

8. Hitung dan gambarlah respon frekuensi dari filter pada nomor 4 dan 6 di atas. Gunakan Matlab.

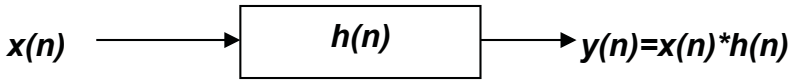
9. Gantilah input pada soal nomor 4 dengan suara dari mikrophone. Gunakan file Simulink `soal10_9.mdl`. Amati efek dari filter tersebut terhadap suara anda.

Catatan: Semua file `*.mdl` dan `*.mat` dapat diunduh di situs buku ini.

12

Desain Filter FIR dengan Metode Sampling Frekuensi

Pada Bab 12 sampai Bab 14, kita akan mempelajari beberapa metode untuk mendesain filter FIR. Filter FIR pada umumnya dinyatakan dalam bentuk *impulse response* seperti pada Gambar 12.1, sehingga hasil akhir dari proses desain suatu filter FIR adalah nilai dari *h(n)*.



Gambar 12.1
Filter FIR dalam bentuk impulse respons

Filter FIR dapat didesain dengan banyak metode tetapi pada buku ini hanya akan dibahas tiga metode yaitu metode Sampling Frekuensi (Bab 12), metode Window (Bab 13), dan metode Optimal (Bab 14). Metode yang diajarkan dalam bab-bab ini bertujuan untuk memberikan pengetahuan dasar

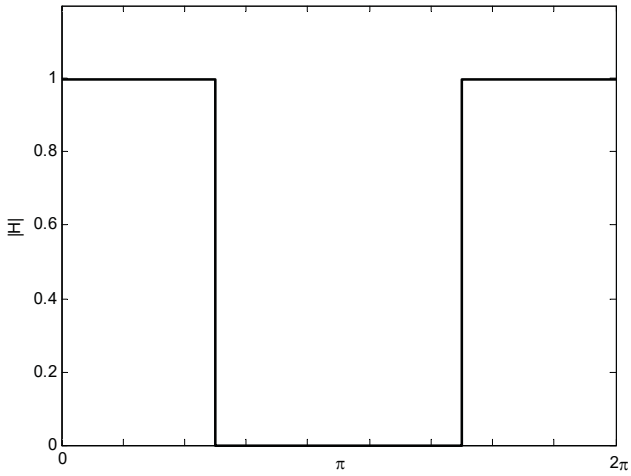
tentang cara menghitung *impulse response*, *h(n)*, dari filter FIR secara manual. Dalam aplikasi praktisnya *h(n)* tidak akan selalu dihitung secara manual. Desain filter FIR akan dilakukan dengan menggunakan program komputer karena nilai *h(n)* yang harus dihitung akan sangat banyak. Ada banyak program komputer yang dapat digunakan untuk menghitung *h(n)* dari FIR. Pada modul-modul ini kita akan mempelajari juga cara menggunakan Matlab untuk mendesain filter FIR.

Setelah mempelajari materi pada bab ini maka mahasiswa akan dapat mendesain filter FIR dengan metode sampling frekuensi, baik secara perhitungan manual maupun dengan menggunakan perangkat lunak pendukung.

12.1 Metode Sampling Frekuensi

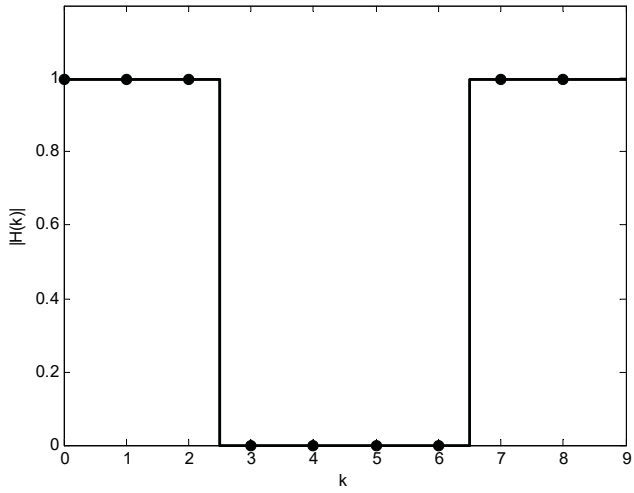
Metode ini adalah metode yang sederhana tetapi menghasilkan respon frekuensi yang tidak terlalu sempurna karena kita tidak dapat dengan mudah menentukan *ripple* baik pada *passband* maupun *stopband*. Metode ini menggunakan langkah-langkah yang akan dijelaskan dengan menggunakan contoh berikut ini dimana kita akan merancang LPF filter dengan orde, $N = 8$, dan $\Omega_c = 5/9\ \pi$. Langkah-langkah tersebut adalah sebagai berikut:

1. Gambarlah respon frekuensi ideal dari filter yang akan dibuat dari 0 sampai 2π. Gambar tersebut dimulai dari |H| = 1 sampai titik $5/9\ \pi$ kemudian turun menjadi |H|=0 sampai π. Gambar sesudah $\Omega = \pi$. adalah pencerminan dari respon frekuensi dari $\Omega = 0$ sampai $\Omega = \pi$.



Gambar 12.2
Respons frekuensi ideal dari LPF dengan $\Omega_c = 5/9\ \pi$.

2. Tandailah sebanyak *N+1* titik sample (dengan interval frekuensi yang sama) pada gambar respons frekuensi tersebut dimana *N* adalah orde dari filter yang akan dibuat. Karena kita akan merancang filter dengan orde, N=8, maka kita menandai 9 titik seperti pada Gambar 12.3 di bawah ini. Catatlah nilai dari sample-sample tersebut. Titik-titik sample ini disebut *H(k)* untuk *k = 0 ... N-1*.



Gambar 12.3
Titik sample pada respons frekuensi LPF dengan $\Omega_c = 5/9\ \pi$.

Nilai *H(k)* adalah sebagai berikut:

H(0) = 1
H(1) = 1
H(2) = 1
H(3) = 0
H(4) = 0
H(5) = 0
H(6) = 0
H(7) = 1
H(8) = 1

3. Hitunglah *h(n)* dengan menggunakan

$$h(n) = \frac{1}{2M+1}\left\{H(0) + 2\sum_{k=1}^{M} H(k)\,\cos\left(\frac{2\pi k(n-M)}{2M+1}\right)\right\}$$

dimana *M =½ N*, jadi untuk contoh ini, *M = ½ (8) = 4*

maka

$$h(n) = \frac{1}{9}\left\{1 + 2\sum_{k=1}^{4} H(k)\ \cos\left(\frac{2\pi k(n-4)}{9}\right)\right\}$$

sehingga

$$h(0) = \frac{1}{9}\left\{1 + 2\left(H(1)\ \cos\left(\frac{2\pi 1(0-4)}{9}\right) + H(2)\ \cos\left(\frac{2\pi 2(0-4)}{9}\right)\right)\right\}$$

$$h(0) = \frac{1}{9}\left\{1 + 2\left(\cos\left(\frac{-8\pi}{9}\right) + \cos\left(\frac{-16\pi}{9}\right)\right)\right\} = 0.0725$$

$$h(1) = \frac{1}{9}\left\{1 + 2\left(\cos\left(\frac{-6\pi}{9}\right) + \cos\left(\frac{-12\pi}{9}\right)\right)\right\} = -0.1111$$

$$h(2) = \frac{1}{9}\left\{1 + 2\left(\cos\left(\frac{-4\pi}{9}\right) + \cos\left(\frac{-8\pi}{9}\right)\right)\right\} = -0.0591$$

$$h(3) = \frac{1}{9}\left\{1 + 2\left(\cos\left(\frac{-2\pi}{9}\right) + \cos\left(\frac{-4\pi}{9}\right)\right)\right\} = 0.3199$$

$$h(4) = \frac{1}{9}\left\{1 + 2\left(\cos(0) + \cos(0)\right)\right\} = 0.5556$$

$$h(5) = h(3) = 0.3199$$

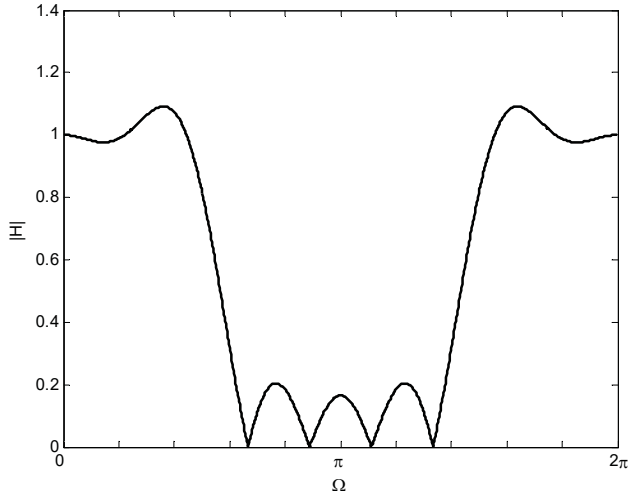
$$h(6) = h(2) = -0.0591$$

$$h(7) = h(1) = -0.1111$$

$$h(8) = h(0) = 0.0725$$

Dengan melakukan 3 langkah di atas maka kita telah selesai mendesain filter FIR orde 8 dengan $\Omega_c = 5/9\ \pi$.

Untuk memastikan respon frekuensi dari filter yang baru didesain, maka *h(n)* tersebut ditransformasi ke domain frekuensi dengan FFT dengan bantuan zero padding (lihat Bab 11) dan hasilnya ditunjukkan pada Gambar 12.4.



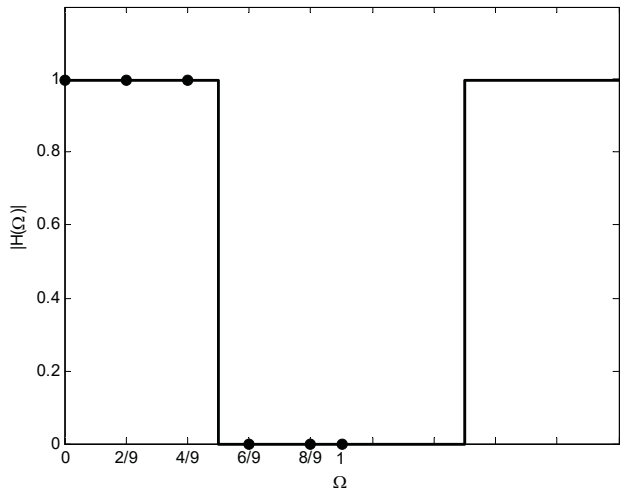
Gambar 12.4
Respons frekuensi dari LPF dengan $\Omega_c = 5/9\ \pi$.

Gambar 12.4 menunjukkan bahwa filter hasil desain ini memiliki respon frekuensi yang tidak persis sama dengan respon frekuensi yang diharapkan pada Gambar 12.2. Terdapat *ripple* baik pada *passband* maupun pada *stopband*. Inilah kelemahan dari metode desain dengan sampling frekuensi. Kita tidak dapat dengan mudah mengatur bersarnya *ripple* dan lebarnya *transition band*. Metode desain ini memeliki ***kelebihan*** yaitu memungkinkan kita merancang filter dengan bentuk respon frekuensi yang sangat bervariasi.
Jika filter FIR yang akan kita rancang memiliki orde yang bernilai gasal maka digunakan rumus:

$$h(n) = \frac{1}{2M+1}\left\{ H(0) + 2\sum_{k=1}^{M-\frac{1}{2}} H(k)\,\cos\left(\frac{2\pi k(n-M)}{2M+1}\right)\right\}$$

12.2 Metode Sampling Frekuensi pada Matlab

Matlab tidak memiliki fungsi untuk mendesain filter FIR dengan metode sampling frekuensi seperti pada langkah-langkah di atas. Fungsi pada Matlab yang paling mirip dengan metode di atas adalah fungsi '`fir2`'. Jika kita menggunakan fungsi ini maka kita harus meletakkan titik sample pada frekuensi Ω = 0 dan Ω = 1 (ingat, untuk Matlab Ω = 1 ini setara dengan $f_s/2$). Fungsi '`fir2`' tidak mengharuskan titik-titik sample diletakkan pada interval frekuensi yang sama sehingga Gambar 12.3 harus digambarkan seperti pada Gambar 12.5.



Gambar 12.5
Titik sample pada respons frekuensi LPF dengan Ω_c = 5/9 π sesuai format Matlab.

Perintah Matlab adalah:

```
>> F=[0  2/9 4/9  6/9  8/9   1];
>> A=[1   1   1    0    0     0];
>> N=8;
>> B=fir2(N,F,A,boxcar(N+1))
```

F adalah frekuensi dari titik-titik sample, *A* adalah nilai dari titik-titik sample tersebut, dan *N* adalah orde dari filter yang dirancang. Output dari perintah tersebut adalah *B* sebagai berikut:

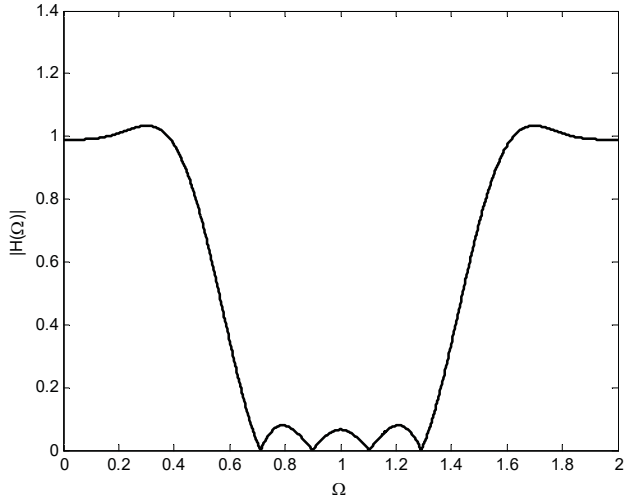
```
B =
  Columns 1 through 6
    0.0363   -0.0762   -0.0503    0.3072    0.5557    0.3072

  Columns 7 through 9
   -0.0503   -0.0762    0.0363
```

maka

h(0) = 0.0363
h(1) = -0.0762
h(2) = -0.0503
h(3) = 0.3072
h(4) = 0.5557
h(5) = 0.3072
h(6) = -0.0503
h(7) = -0.0762
h(8) = 0.0363

Nilai ini berbeda dengan hasil perhitungan manual di atas karena Matlab menuntut kita untuk harus menyelipkan satu titik pada $\Omega = 1$ yang setara dengan $f_s/2$.

Untuk memastikan respon frekuensi dari filter yang baru didesain, maka *h(n)* tersebut ditransformasi ke domain frekuensi dengan FFT seperti pada Gambar 12.6.



Gambar 12.6
Respons frekuensi LPF dengan $\Omega_c = 5/9\ \pi$ yang didesain dengan Matlab.

SOAL LATIHAN

1. Rancanglah suatu filter HPF orde *8* dengan $\Omega_c = 4/9\ \pi$ dengan metode sampling frekuensi.

2. Ulangi soal nomor 2 di atas dengan menggunakan Matlab.

3. Rancanglah suatu filter LPF orde *9* dengan $f_c = 2\ KHz$, $f_{stop} = 3\ KHz$ dan $f_s = 10\ kHz$ dengan metode sampling frekuensi.

4. Ulangi soal nomor 3 di atas dengan menggunakan Matlab.

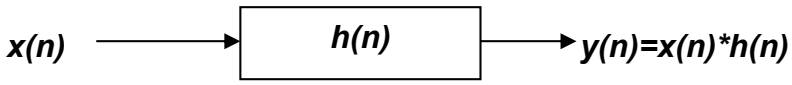
5. Implementasikan filter tersebut dengan memodifikasi file `soal11_5.m`. Berikan beberapa sinyal sinusoidal dengan frekuensi *500 Hz, 1 kHz, 2 kHz, 4 kHz* dan *4 kHz* (gunakan $f_s = 10\ kHz$) sebagai inputnya. Amati apakah filter tersebut berfungsi sebagaimana mestinya.

6. Matlab memiliki fungsi 'filter' seperti terlihat pada file `soal11_6.m`. Ulangi soal nomor 5 di atas dengan menggunakan fungsi filter. Amati apakah filter tersebut berfungsi sebagaimana mestinya.

7. Implementasikan filter di atas dengan menggunakan Simulink untuk sinyal input seperti pada soal nomor 5.

13
Desain Filter FIR dengan Metode Window

Pada bab ini, kita akan mempelajari metode untuk mendesain filter FIR dengan mengunakan metode window. Sama seperti pada bab sebelumnya, hasil akhir dari proses desain suatu filter FIR dengan metode window adalah nilai *h(n)* dari filter FIR seperti pada Gambar 13.1.



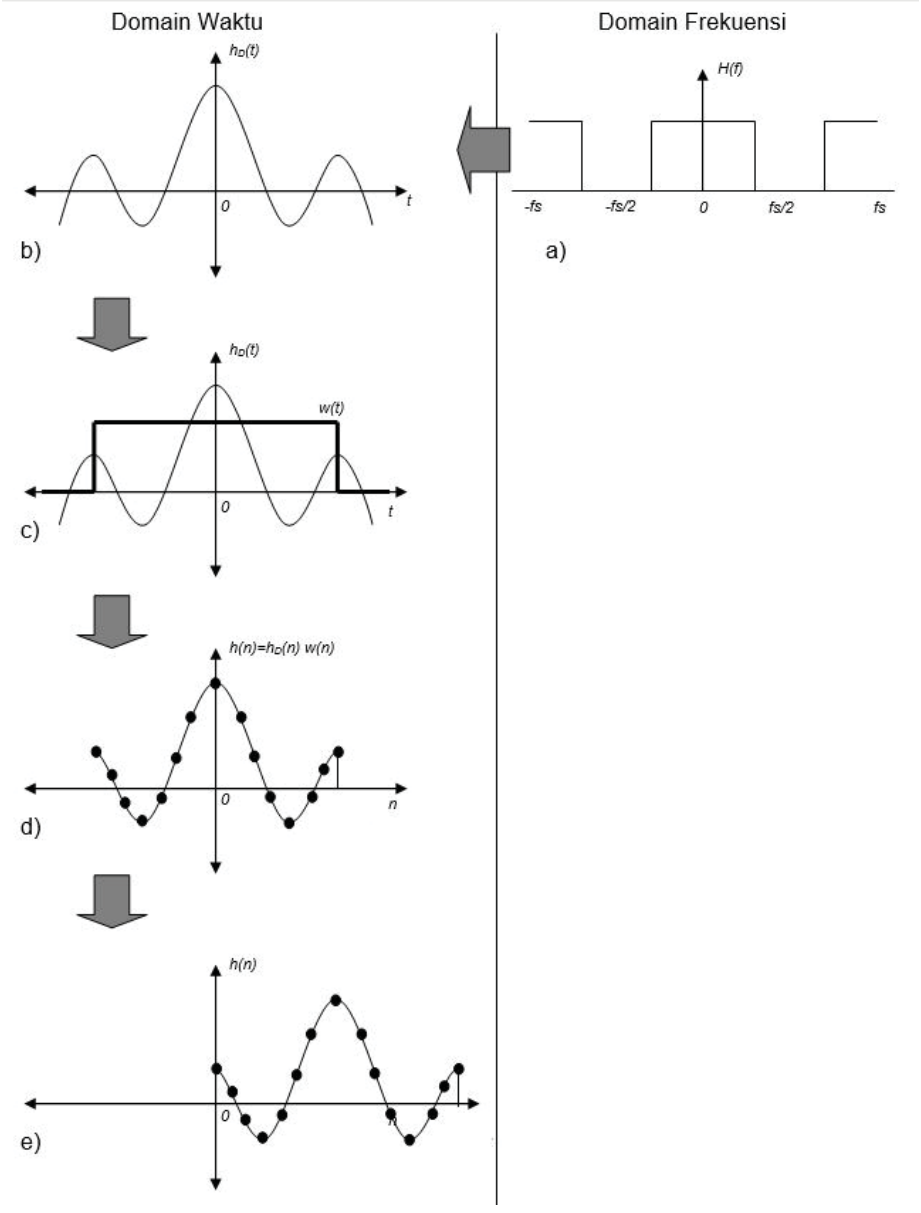
Gambar 13.1
Filter FIR dalam bentuk impulse respons

Setelah mempelajari materi pada bab ini mahasiswa akan dapat mendesain filter FIR dengan metode window, baik secara perhitungan manual maupun dengan menggunakan perangkat lunak pendukung.

13.1 Proses Desain dengan Metode Window

Prinsip dasar dari metode desain ini digambarkan pada Gambar 13.2. Proses desain dimulai dari menggambarkan dan menulis persamaan response frekuensi, *H(f)*, dari filter yang akan dibuat dari $-f_s$ sampai f_s. Pada gambar tersebut contoh yang akan dibuat adalah filter LPF (Gambar 13.2a).

Persamaan ini kemudian ditransformasi dengan Inverse Fourier Transform untuk memperoleh *impulse response* ideal, $h_D(t)$. Impulse response ini memiliki panjang yang tak berhingga (Gambar 13.2b). Karena kita akan mendesain filter FIR maka kita harus mempunyai *impulse response* dengan panjang yang terbatas. Untuk membatasi panjang $h_D(t)$, maka *impulse response* tersebut dikali dengan sebuah *window*, $w(t)$. Dalam Gambar 13.2c, digunakan *window rectangular*.

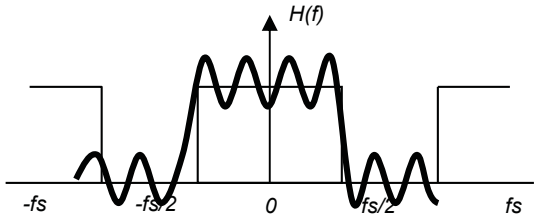


Gambar 13.2
Proses desain FIR dengan metode window

Impulse response yang sudah dibatasi panjangnya ini kemudian disample dengan jumlah titik sesuai dengan orde dari filter yang akan dibuat, *h(n)*, seperti pada Gambar 13.2d. Langkah terakhir adalah menggeser impulse response tersebut agar kita tidak memiliki *h(n)* untuk *n* negatif (Gambar 13.2e). Jika ada *n* yang bernilai negatif maka filter tersebut tidak bersifat causal (bisa menghasilkan output sebelum ada input).

Setelah kita menggeser *h(n)* maka proses desain telah selesai dan kita telah memiliki filter FIR dengan respon frekuensi sesuai yang kita harapkan.

Dampak dari penggunaan *window* terlihat pada saat kita mengamati kembali respon frekuensi dari filter yang sudah dibuat. Ternyata, respon frekuensi tidak sesuai dengan desain awal seperti ditunjukkan pada Gambar 13.3. Terdapat *ripple* dan *transition band* yang lebar.



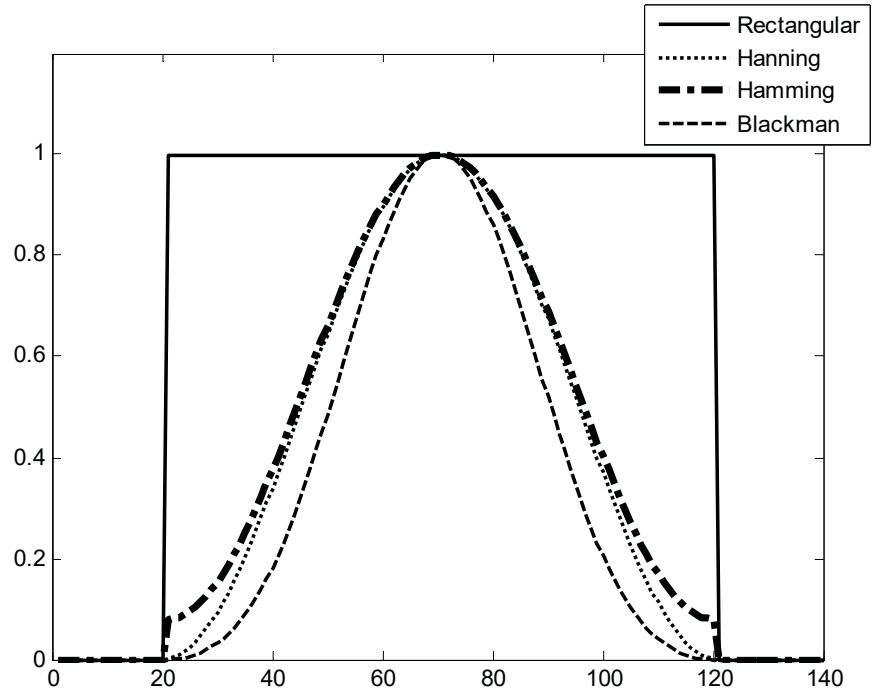
Gambar 13.3
Response frekuensi dari filter dengan rectangular window

Kita dapat mengurangi dampak buruk ini dengan memilih bentuk *window* yang sesuai. Terdapat banyak pilihan *window*, beberapa diantaranya ditunjukkan pada Table 13.1 dan Gambar 13.4

Table 13.1
Berbagai window dan parameternya

Nama Window	Transition Band (Normalized Hz)	Passband ripple (dB)	Stopband attenuation (dB)	Persamaan *w(n)* $\lvert n \rvert \le (N/2)$
Rectangular	0.9/(N+1)	0.7416	21	1
Hanning	3.1/(N+1)	0.0546	44	$0.5+0.5\cos(\frac{2\pi n}{N+1})$
Hamming	3.3/(N+1)	0.0194	53	$0.54+0.46\cos(\frac{2\pi n}{N+1})$
Blackman	5.5/(N+1)	0.0017	74	$0.42+0.5\cos\left(\frac{2\pi n}{N}\right)+0.08\cos\left(\frac{4\pi n}{N}\right)$

N adalah orde dari filter



Gambar 13.4
Bentuk dari window pada Tabel 13.1

Secara sepintas, proses ini terlihat sangat rumit khususnya proses dari Gambar 13.2a ke Gambar 13,2b dimana kita harus melakukan Inverse Fourer Transform (IFT) secara analitik. Untuk memudahkan proses ini maka Tabel 13.2 menuliskan $h_D(n)$ dari beberapa filter yang umum digunakan. Tabel ini menolong kita karena kita tidak perlu melakukan proses IFT.

Table 13.2
Impulse response ideal dari berbagai filter

Tipe Filter	Impulse response ideal, $h_D(n)$	
	$h_D(0)$	$h_D(n),\ n \neq 0$
LPF	$2 f_c$	$2 f_c \frac{\sin(n\omega_c)}{n\omega_c}$
HPF	$1-2 f_c$	$-2 f_c \frac{\sin(n\omega_c)}{n\omega_c}$
BPF	$2 (f_1 - f_2)$	$2 f_2 \frac{\sin(n\omega_2)}{n\omega_2} - 2 f_1 \frac{\sin(n\omega_1)}{n\omega_1}$
BSF	$1-2 (f_1 - f_2)$	$2 f_1 \frac{\sin(n\omega_1)}{n\omega_1} - 2 f_2 \frac{\sin(n\omega_2)}{n\omega_2}$

f_c adalah frekuensi di tengah transition band, demikian juga dengan f_1 dan f_2

Langkah-langkah desain akan ditunjukkan dengan contoh berikut ini. Kita akan mendesain suatu filter:

Tipe filter	: HPF
Frekuensi sampling (f_s)	: 8 kHz
Frekuensi cut-off (fc)	: 1.5 kHz
Transition band	: 500 Hz
Stopband attenuation	: > 50 dB
Passband ripple	: < 0.1 dB

Langkah 1:
Kita akan melihat Tabel 13.2 untuk menemukan persamaan $h_D(n)$ yang sesuai. Karena kita mendesain HPF maka $h_D(n)$ yang sesuai adalah

$$h_D(0) = 1 - 2 f_c$$

$$h_D(n) = -2 f_c \frac{\sin(n\omega_c)}{n\omega_c} \qquad untuk\ n \neq 0$$

Nilai f_c yang digunakan adalah f_c ternormalisasi dan terkoreksi yaitu di tengah-tengah transition band. Karena transition band sebesar *500 Hz*, maka transition band ini berada dari frekuensi *1 kHz* sampai *1.5 kHz*. Nilai f_c' tepat berada di tengah-tengah transition band ini yaitu pada *1.25 kHz*. Nilai f_c ternormalisasi dihitung dengan:

$$f_c = \frac{f'_c}{f_s} = \frac{1.5\,kHz - 0.5(500Hz)}{8\,kHz} = \frac{1.25\ kHz}{8\ kHz} = 0.1563$$

Langkah 2:
Kita akan melihat Tabel 13.1 untuk menemukan persamaan *w(n)* yang sesuai. Karena kita membutuhkan passband ripple < 0.1 dB dan stopband attenuation > 50 dB, maka kita hanya mungkin menggunakan *Hamming window* atau *Blackman window*. Sebaiknya kita memilih *Hamming window* karena memiliki transition band yang lebih sempit.

$$w(n) = 0.54 + 0.46\cos\left(\frac{2\pi n}{N+1}\right)$$

Langkah 3:
Menentukan panjang dari filter. Karena transition band harus 500 Hz maka normalized transition band

$$\Delta f = \frac{transition\ band}{f_s} = \frac{500}{8\,kHz} = 0.0625$$

Normalized transition band dari *Hamming window* adalah 3.3/(N+1) maka

$$0.0625 = \frac{3.3}{N+1}$$

Sehingga $N+1 = 52.8$ (atau dibulatkan menjadi *53*), sehingga $N = 52$. Artinya untuk memenuhi parameter dari filter ini, kita harus mendesain filter dengan orde *52*.

Langkah 4:
Menghitung $h(n) = h_D(n)\, w(n)$. Karena kita akan mendesain filter dengan orde *52* maka kita akan menghitung *53* buah nilai $h(n)$ untuk $-26 \leq n \leq 26$.

Kita cukup menghitung $h(n)$ untuk $0 \leq n \leq 26$. Nilai $h(n)$ untuk n negatif adalah pencerminan dari nilai $h(n)$ untuk n positif yaitu

$h(-1) = h(1)$
$h(-2) = h(2)$
...
$h(-26) = h(26)$

Nilai nilai $h(n)$ adalah:

$$h(0) = (1-2f_c)\left(0.54+0.46\cos\left(\frac{2\pi 0}{53}\right)\right) = (1-2(0.1563))(0.54+0.46) = 0.6874$$

$$h(1) = \left(-2f_c\frac{\sin(n\omega_c)}{n\omega_c}\right)\left(0.54+0.46\cos\left(\frac{2\pi 1}{53}\right)\right) = \left(-2(0.1563)\frac{\sin(1\,2\pi 0.1563)}{1\,(2\pi 0.1563)}\right)\left(0.54+0.46\cos\left(\frac{2\pi}{53}\right)\right) = -0.2639$$

$$h(2) = \left(-2f_c\frac{\sin(n\omega_c)}{n\omega_c}\right)\left(0.54+0.46\cos\left(\frac{2\pi 2}{53}\right)\right) = \left(-2(0.1563)\frac{\sin(2\,2\pi 0.1563)}{2\,(2\pi 0.1563)}\right)\left(0.54+0.46\cos\left(\frac{2\pi 2}{53}\right)\right) = -0.1451$$

...

$$h(26) = \left(-2f_c\frac{\sin(n\omega_c)}{n\omega_c}\right)\left(0.54+0.46\cos\left(\frac{2\pi 26}{53}\right)\right) = \left(-2(0.1563)\frac{\sin(26\,2\pi 0.1563)}{26\,(2\pi 0.1563)}\right)\left(0.54+0.46\cos\left(\frac{2\pi 26}{53}\right)\right) = -0.0004$$

Nilai-nilai ini akan sangat sulit dihitung secara manual. File `Fig13_5.m` pada modul ini dapat digunakan untuk menghitung semua nilai *h(n)*. Hasil perhitunganya ditunjukkan pada Tabel 13.3.

Table 13.3
Impulse response dari HPF

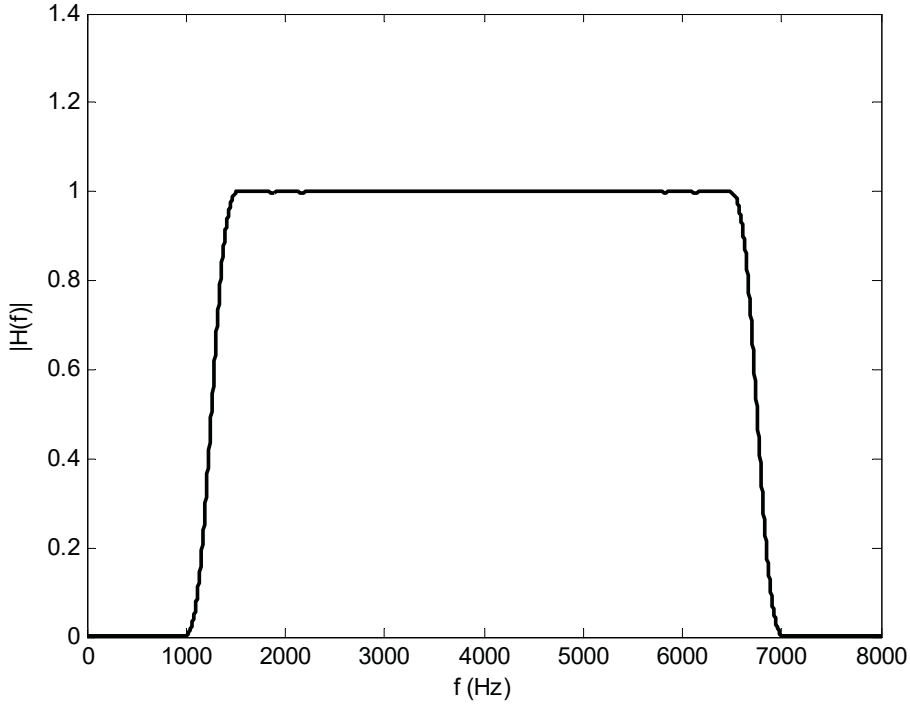
	0.6874	= h(0)
h(-1) =	-0.2639	= h(1)
h(-2) =	-0.1451	= h(2)
h(-3) =	-0.0200	= h(3)
h(-4) =	0.0535	= h(4)
h(-5) =	0.0575	= h(5)
h(-6) =	0.0180	= h(6)
h(-7) =	-0.0216	= h(7)
h(-8) =	-0.0322	= h(8)
h(-9) =	-0.0149	= h(9)
h(-10) =	0.0087	= h(10)
h(-11) =	0.0188	= h(11)
h(-12) =	0.0114	= h(12)
h(-13) =	-0.0027	= h(13)
h(-14) =	-0.0105	= h(14)
h(-15) =	-0.0078	= h(15)
h(-16) =	0.0000	= h(16)
h(-17) =	0.0053	= h(17)
h(-18) =	0.0048	= h(18)
h(-19) =	0.0008	= h(19)
h(-20) =	-0.0024	= h(20)
h(-21) =	-0.0026	= h(21)
h(-22) =	-0.0008	= h(22)
h(-23) =	0.0009	= h(23)
h(-24) =	0.0013	= h(24)
h(-25) =	0.0006	= h(25)
h(-26) =	-0.0004	= h(26)

Langkah 5:
Menggeser impulse response agar semua *n* bernilai positif sehingga filter bersifat causal. Proses ini dilakukan dengan mengganti semua *n* dengan *n+26* seperti pada Tabel 13.4.

Table 13.4
Impulse response dari HPF yang bersifat causal

	0.6874	= h(26)
h(25) =	-0.2639	= h(27)
h(24) =	-0.1451	= h(28)
h(23) =	-0.0200	= h(29)
h(22) =	0.0535	= h(30)
h(21) =	0.0575	= h(31)
h(20) =	0.0180	= h(32)
h(19) =	-0.0216	= h(33)
h(18) =	-0.0322	= h(34)
h(17) =	-0.0149	= h(35)
h(16) =	0.0087	= h(36)
h(15) =	0.0188	= h(37)
h(14) =	0.0114	= h(38)
h(13) =	-0.0027	= h(39)
h(12) =	-0.0105	= h(40)
h(11) =	-0.0078	= h(41)
h(10) =	0.0000	= h(42)
h(9) =	0.0053	= h(43)
h(8) =	0.0048	= h(44)
h(7) =	0.0008	= h(45)
h(6) =	-0.0024	= h(46)
h(5) =	-0.0026	= h(47)
h(4) =	-0.0008	= h(48)
h(3) =	0.0009	= h(49)
h(2) =	0.0013	= h(50)
h(1) =	0.0006	= h(51)
h(0) =	-0.0004	= h(52)

Untuk memastikan respon frekuensi dari filter yang baru didesain, maka *h(n)* tersebut ditransformasi ke domain frekuensi dengan FFT dan hasilnya ditunjukkan pada Gambar 13.5. Gambar tersebut menunjukkan bahwa filter yang dirancang memiliki respon frekuensi sesuai spesifikasi desain.



Gambar 13.5
Respons frekuensi dari HPF yang didesain

13.2 Metode Window pada Matlab

Untuk mendesain FIR dengan metode window pada Matlab kita gunakan fungsi 'fir1'. Untuk filter yang didesain di atas kita menggunakan perintah:

```
>> h=fir1(52,1250/4000,'high',hamming(53))
```

Angka *52* adalah orde dari filter yang dirancang, *1250* adalah frekuensi cut-off terkoreksi, f_c', *4000* adalah setengah f_s, dan window hamming dengan panjang *53*.

SOAL LATIHAN

1. Dengan menggunakan metode *window* rancanglah filter dengan spesifikasi sebagai berikut:

Tipe filter	: LPF
Frekuensi sampling (f_s)	: 10 kHz
Frekuensi cut-off (fc)	: 2 kHz
Transition band	: 1 kHz
Stopband attenuation	: > 25 dB
Passband ripple	: < 0.1 dB

   Hitunglah nilai *h(0)*, *h(N/2)*, dan *h(N)* saja.

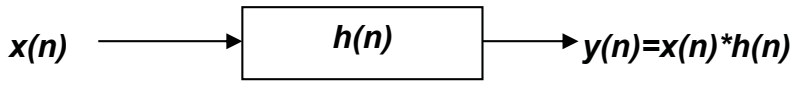
2. Gunakan fungsi '`fir1`' pada Matlab untuk menghitung semua *h(n)* dari filter pada soal nomor 1.

3. Dengan memodifikasi file `modul13_3.m`, gambarlah respon frekuensi dari filter yang didesain di soal nomor 2.

4. Dengan memodifikasi file `modul13_4.m`, gunakan filter yang didesain tersebut untuk berbagai input sinusoidal untuk memeriksa apakah filter tersebut berfungsi dengan benar.

14
Desain Filter FIR dengan Metode Optimal

Pada bab ini, kita akan mempelajari cara untuk mendesain filter FIR dengan mengunakan metode optimal. Sama seperti pada bab sebelumnya, hasil akhir dari proses desain suatu filter FIR dengan metode optimal adalah nilai *h(n)* dari filter FIR seperti pada Gambar 14.1.



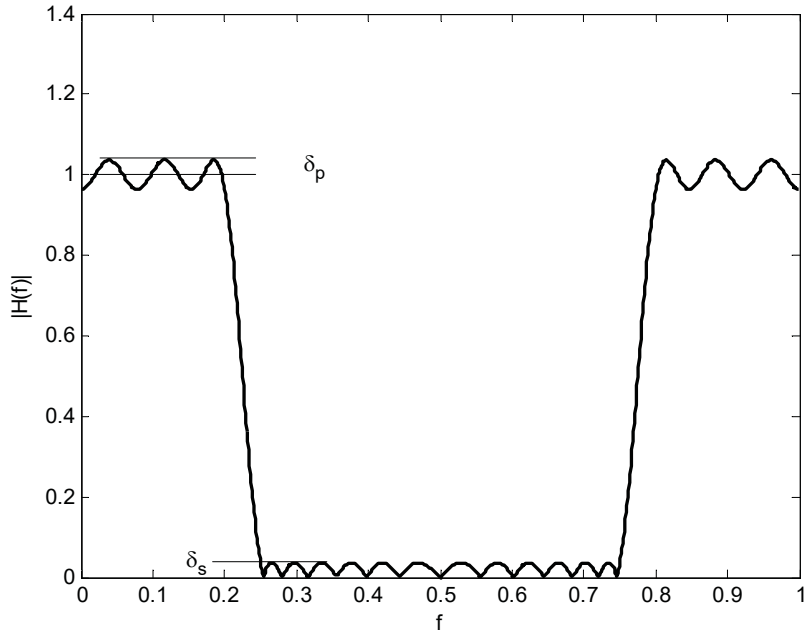
Gambar 14.1
Filter FIR dalam bentuk impulse respons

Setelah mempelajari materi pada bab ini mahasiswa akan dapat mendesain filter FIR dengan metode optimal, baik secara perhitungan manual maupun dengan menggunakan perangkat lunak pendukung.

Metode ini menghasilkan filter yang sangat baik sehingga menjadi pilihan pertama dalam setiap desain FIR. Metode ini

menggunakan proses komputasi yang sangat panjang sehingga lebih praktis untuk diterapkan dengan bantuan komputer.

Kedua metode terdahulu selalu menghasilkan filter dengan *ripple* baik di *passband* maupun di *stopband*. Kita juga melihat bahwa *ripple* akan semakin besar di daerah sekitar frekuensi cut-off. Metode optimal ini berusaha untuk membuat filter FIR dengan *ripple* yang memiliki amplitodo konstan (*equiripple*) seperti ditunjukkan pada Gambar 14.2. Dengan demikian maka kita akan mendapat filter dengan respon frekuensi yang paling optimal.



Gambar 14.2
Filter FIR dengan equiripple

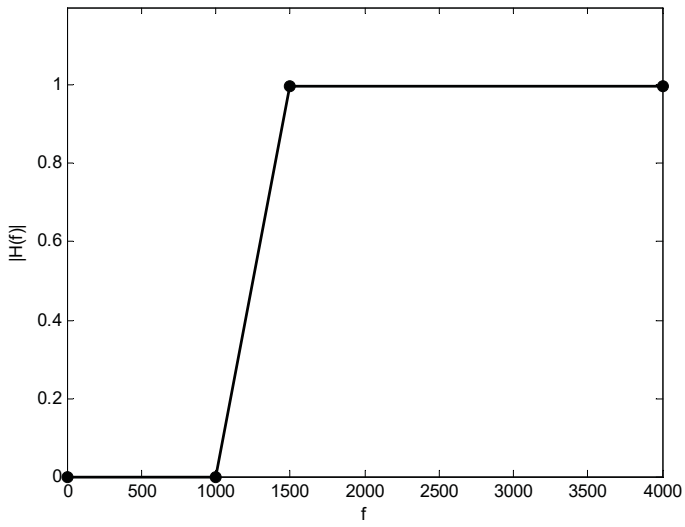
Metode ini berusaha untuk meminimalisasi error, $E(\omega)$, yaitu selisih antara respon frekuensi ideal, $H_D(\omega)$, dengan respon frekuensi hasil desain, $H(\omega)$ setelah dikalikan dengan fungsi weighting, $W(\omega)$.

$$E(\omega) = W(\omega)[H_D(\omega) - H(\omega)]$$

Langkah-langkah desain akan ditunjukkan dengan contoh berikut ini. Kita akan mendesain suatu filter

Tipe filter	: HPF
Frekuensi sampling (f_s)	: 8 kHz
Frekuensi cut-off (f_c)	: 1.5 kHz
Transition band	: 500 Hz
Stopband attenuation	: > 40 dB
Passband ripple	: < 0.1 dB
Orde	: 40

Langkah 1:
Untuk memudahkan proses, kita akan menggambar respon frekuensi ideal yang diharapkan sampai dengan $f_s/2$. Kita menandai titik-titik sudut dari gambar tersebut.



Gambar 14.3
Respon krekuensi ideal dari filter yang akan didesain

Langkah 2:
Catat frekuensi dan amplotudo dari titik-titik sudut tersebut. Kita gunakan frekuensi yang sudah dinormalisasi ke $f_s/2$.

F = [0 1000/4000 1500/4000 4000/4000]

dan

A = [0 0 1 1]

Langkah 3:
Mencari perbandingan δp dan δs. Pada filter ini

$$\delta p = 0.1\ dB = 10^{\left(\frac{0.1}{20}\right)} = 1.0116$$

$$\delta s = -40\ dB = 10^{\left(\frac{-40}{20}\right)} = 0.01$$

Jadi perbandingan

$$\delta p : \delta s = 101 : 1$$

maka kita membuat

W = [1 101]

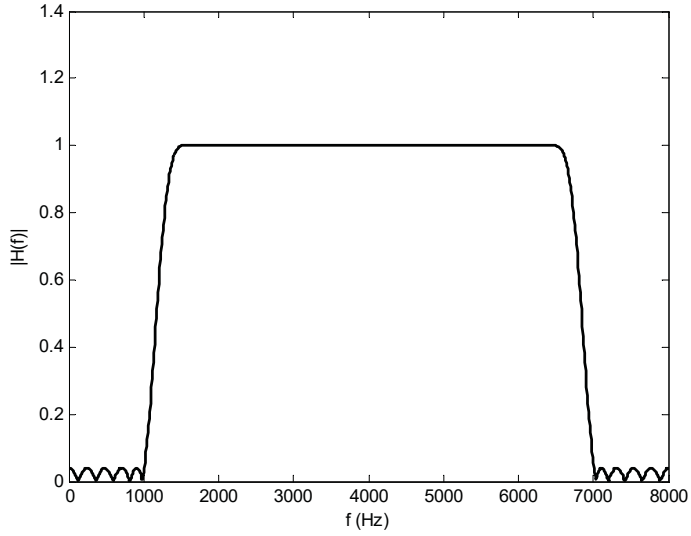
Karena filter yang dibuat adalah HPF maka nilai W dimulai dari rasio *stopband* (1) diikuti rasio *passband* (101).

Langkah 4:
Menggunakan perintah Matlab untuk menghitung h(n).

```
>> h=firpm(40,F,A,W);
```

Detail proses desain dapat dilihat di file `Fig13_4.m`.

Untuk memastikan respon frekuensi dari filter yang baru didesain, maka *h(n)* tersebut ditransformasi ke domain frekuensi dengan FFT dan hasilnya ditunjukkan pada Gambar 14.4. Gambar tersebut menunjukkan bahwa filter yang dirancang memiliki respon frekuensi sesuai spesifikasi desain.



Gambar 14.4
Respon frekuensi dari filter hasil desain.

SOAL LATIHAN

1. Dengan menggunakan metode optimal rancanglah filter dengan spesifikasi sebagai berikut:

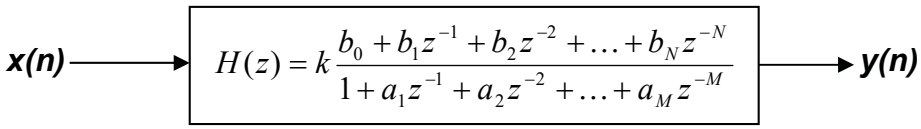
Tipe filter	: LPF
Frekuensi sampling (f_s)	: 10 kHz
Frekuensi cut-off (fc)	: 2 kHz
Transition band	: 1 kHz
Stopband attenuation	: > 25 dB
Passband ripple	: < 0.1 dB
Orde	: 30

2. Dengan memodifikasi file `Fig14_4.m`, gambarlah respon frekuensi dari filter yang didesain di soal nomor 1.

3. Dengan memodifikasi file `modul14_3.m`, gunakan filter yang didesain tersebut untuk berbagai input sinusoidal untuk memeriksa apakah filter tersebut berfungsi dengan benar.

15
Desain Filter IIR dengan Metode Pole-Zero Placement

Setelah kita mempelajari cara mendesain filter FIR maka pada Bab 15 sampai Bab 17, kita akan mempelajari beberapa metode untuk mendesain filter IIR. Filter IIR memiliki *impulse response* yang tak berhingga sehingga biasanya dinyatakan dalam bentuk *transfer function*, *H(z)*, seperti pada Gambar 15.1. Hasil akhir dari proses desain suatu filter IIR adalah nilai dari koefisien filter b_n, a_n, dan gain, k.



Gambar 15.1
Filter IIR dalam bentuk transfer function

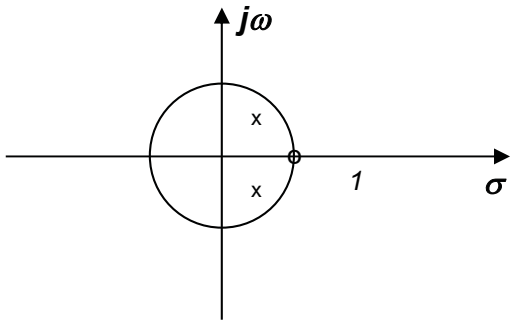
Ada banyak metode untuk mendesain filter IIR. Pada bab-bab ini hanya akan dibahas tiga metode yaitu metode pole-zero placement (Bab 15), metode transformasi bilinier (Bab 16),

dan metode impulse invariant (Bab 17). Metode yang diajarkan dalam bab-bab ini bertujuan untuk memberikan pengetahuan dasar tentang cara mendesain filter IIR secara manual. Sama seperti pada filter FIR, dalam aplikasi praktisnya filter-filter ini tidak akan selalu dihitung secara manual. Desain filter IIR akan dilakukan dengan menggunakan program komputer. Pada bab-bab ini kita juga akan mempelajari juga cara menggunakan Matlab untuk mendesain filter IIR.

Setelah mempelajari materi pada bab ini mahasiswa akan dapat mendesain filter IIR dengan metode *pole-zero placement*, baik secara perhitungan manual maupun dengan menggunakan perangkat lunak pendukung.

15.1 Metode Pole-Zero Placement

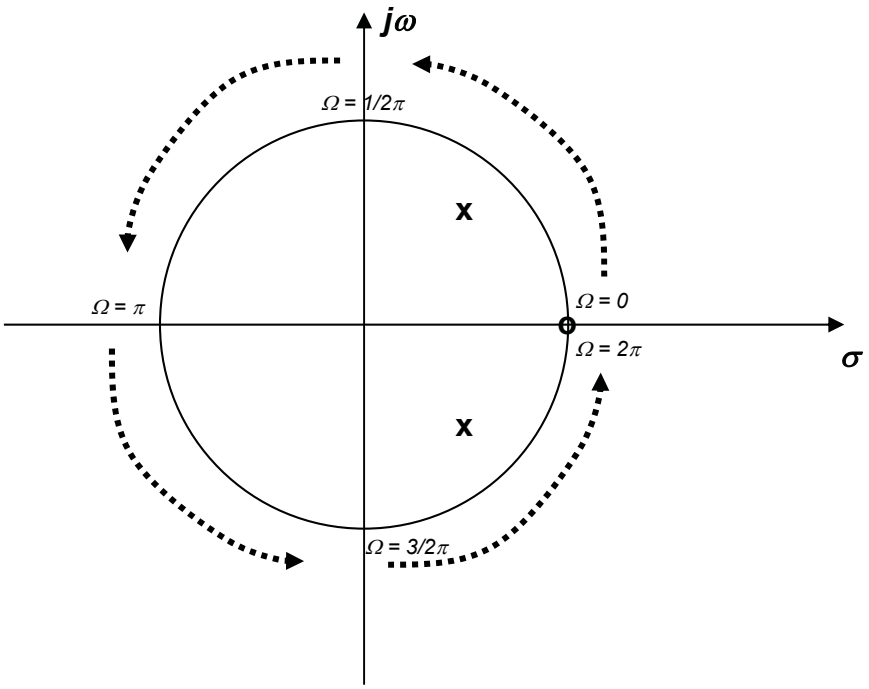
Metode ini adalah metode yang sederhana sehingga mudah digunakan tetapi menghasilkan respon frekuensi yang tidak terlalu presisi. Metode ini didasarkan pada letak dari pole dan zero pada bidang z seperti ditunjukkan pada Gambar 15.2. Bidang ini terdiri dari sumbu real dan imajiner serta lingkaran dengan jari-jari 1.



Gambar 15.2
Pole dan zero pada bidang z

Posisi zero, 'o', menunjukkan bahwa pada posisi itu *H(z)* bernilai nol sedangkan posisi pole, 'x', menunjukkan bahwa pada posisi itu *H(z)* bernilai sangat besar.

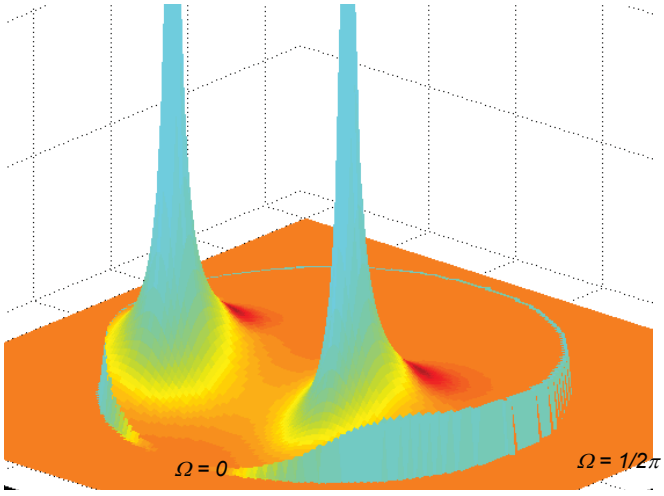
Hubungan antara posisi pole-zero dengan respons frekuensi dapat dilihat pada penampang luar dari lingkaran berjari-jari satu tersebut dimulai dari frekuensi *0* sampai $\pi/2$ pada sumbu $j\omega$, kemudian frekuensi π pada sumbu real negatif, $3/2\ \pi$ pada sumbu $-j\omega$ dan kembali ke nol seperti ditunjukkan pada Gambar 15.3.



Gambar 15.3
Hubungan pole-zero dengan respons frekuensi

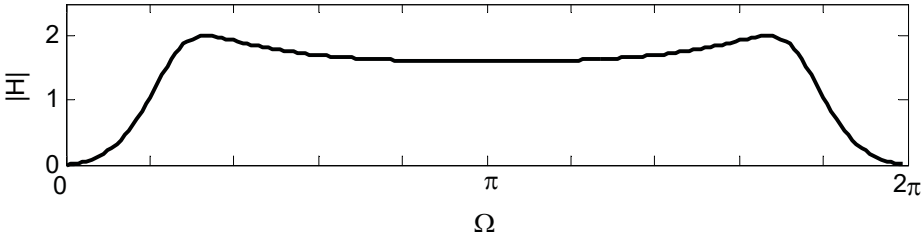
Bentuk penampang luar dari lingkaran dengan jari-jari 1 tersebut adalah seperti pada Gambar 15.4. Penampang luar

ini menunjukkan respons frekuensi dari filter dengan posisi pole-zero seperti pada Gambar 15.3.



Gambar 15.4
Penampang luar dari lingkaran dengan jari-jari satu pada bidang z

Dari Gambar 15.4 kita dapat melihat bahwa filter dengan posisi pole-zero seperti pada Gambar 15.3 adalah filter HPF. Respon frekuensinya digambarkan pada Gambar 15.5.



Gambar 15.5
Respon frekuensi dari filter dengan posisi pole-zero seperti pada Gambar 15.3

Dengan mengatur letak dari pole dan zero kita dapat membuat filter IIR dengan mudah. Untuk menjamin bahwa filter kita selalu memiliki nilai real maka pole dan zero juga harus memiliki nilai real atau dibuat dalam bentuk berpasangan (*complex-conjugate*).

15.2 Contoh Proses Desain

Sebagai contoh, kita akan membuat suatu filter LPF dengan f_c *= 1 kHz*, f_s *= 10 kHz*, dengan transition band *500 Hz*. Karena transition band adalah *500 Hz* maka kita menempatkan zero pada 1.5 kHz yang setara dengan sudut 54^0 tepat pada lingkaran dengan jari-jari 1. Kita akan meletakkan pole pada sudut 36^0 yang setara dengan *1 kHz* tetapi pada jarak *0.8* dari pusat lingkaran agar *ripplenya* tidak terlalu besar seperti ditunjukkan pada Gambar 15.6.

Pole dan zero dari filter ini adalah:

P1 = 0.6472 + j 0.4702
P2 = 0.6472 - j 0.4702

Z1= 0.5878 + j 0.8090
Z2= 0.5878 - j 0.8090

Persamaan dari filter yang dibuat adalah:

$$H(z) = k\frac{(z-(0.5878+j0.8090))(z-(0.5878-j0.8090))}{(z-(0.6472+j0.4702))(z-(0.6472-j0.4702))}$$

atau sama dengan

$$H(z) = k\frac{1-1.1756\,z^{-1}+z^{-2}}{1-1.2944\,z^{-1}+0.64\,z^{-2}}$$

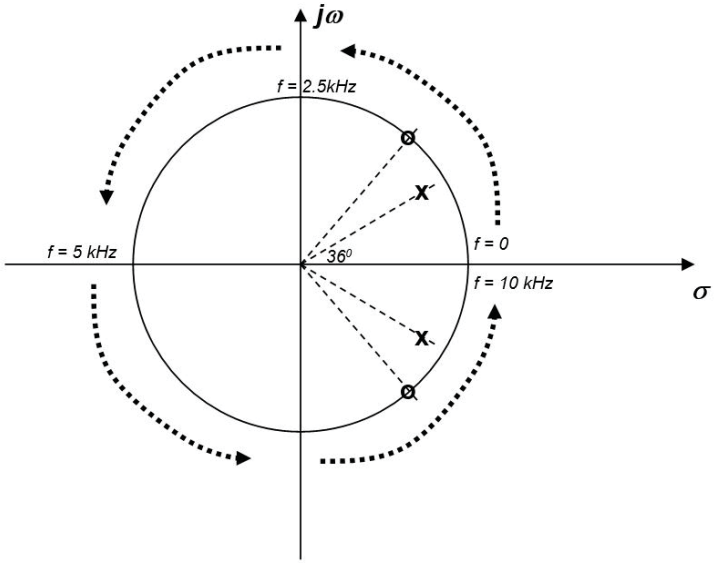
Karena filter yang dirancang adalah LPF maka faktor pengali, *k*, ditentukan agar penguatan, *H(z) = 1* pada *f = 0 Hz* (setara *z = 1*), jadi:

$$1 = k\,\frac{1 - 1.1756 + 1}{1 - 1.2944 + 0.64}$$

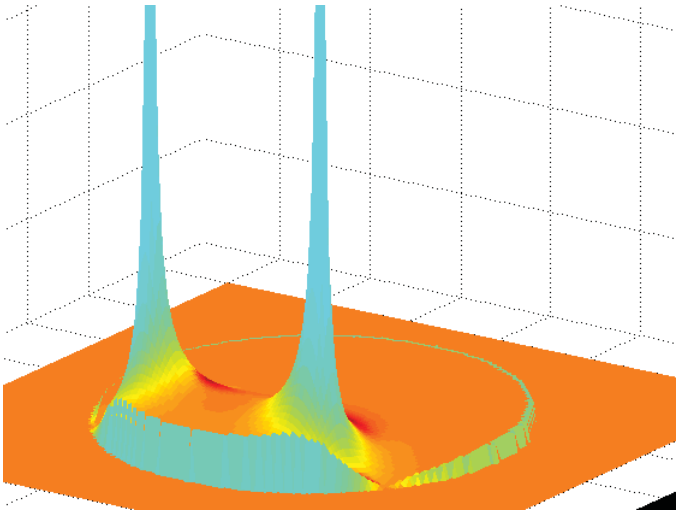
maka *k = 0.4129*, sehingga

$$H(z) = 0.4129\,\frac{1 - 1.1756\,z^{-1} + z^{-2}}{1 - 1.2944\,z^{-1} + 0.64\,z^{-2}}$$

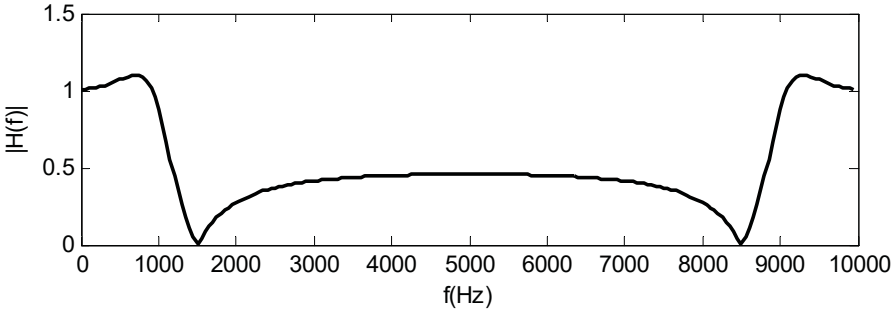
Jika kita merancang HPF maka *k* ditentukan dengan *H(z) = 1* pada *f = f_s/2* (setara *z = -1*). Untuk BPF *H(z) = 1* pada saat *f = 0.5*(f_2-f_1)*



Gambar 15.6
Posisi pole-zero dari LPF dengan f_c = 1 kHz, f_s = 10 kHz dan transition band 500 Hz



Gambar 15.7
Penampang liar lingkaran dari LPF dengan pole-zero seperti pada Gambar 15.6



Gambar 15.8
Respons frekuensi dari LPF dengan pole-zero seperti pada Gambar 15.6

Terlihat bahwa filter ini sudah sesuai dengan spesifikasi desainnya tetapi masih memiliki *ripple* pada *stopband* yang terlalu besar.

SOAL LATIHAN

1. Dengan metode pole-zero placement, rancanglah suatu filter LPF dengan $f_c = 1.5$ *kHz*, $f_s = 8$ *kHz*.

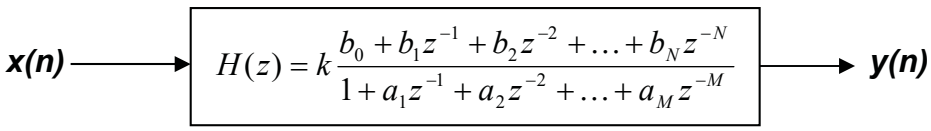
2. Implementasikan filter tersebut dengan Matlab. Berikan beberapa sinyal sinusoidal dengan frekuensi *500 Hz, 1 kHz, 2 kHz*, dan *2.5 kHz* (gunakan f_s = 8 kHz) sebagai inputnya. Amati apakah filter tersebut berfungsi sebagaimana mestinya.

3. Implementasikan filter di atas dengan menggunakan Simulink untuk sinyal input seperti pada soal nomor 5.

16

Desain Filter IIR dengan Metode Transformasi Bilinier

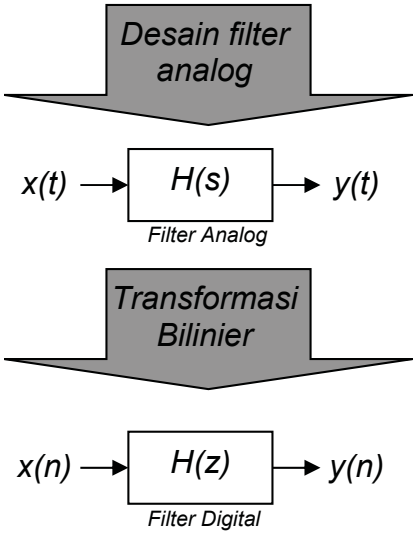
Pada bab ini kita akan mempelajari cara mendesain filter IIR dengan metode transformasi bilinier. Filter IIR biasanya dinyatakan dalam bentuk transfer function, *H(z)*, seperti pada Gambar 16.1. Hasil akhir dari proses desain suatu filter IIR dengan metode transformasi bilinier adalah nilai dari koefisien filter b_n, a_n, dan gain, k.



Gambar 16.1
Filter IIR dalam bentuk transfer function

Metode desain filter IIR dengan transformasi bilinier ini membuat filter digital dengan cara mentransformasi persamaan *H(s)* dari filter analog, maka proses desain filter IIR dengan metode transformasi bilinier dimulai dari proses mendesain filter analog. Filter analog ini kemudian

ditransformasi menjadi filter digital dengan metode transformasi bilinier, seperti ditunjukkan pada Gambar 16.2.

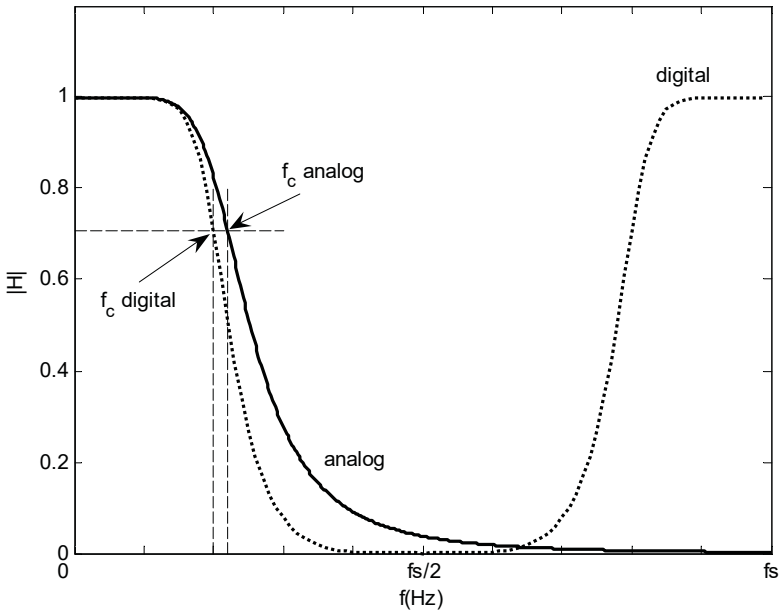


Gambar 16.2
Proses desain filter IIR dengan metode transformasi bilinier

Karena proses desainnya dimulai dengan filter analog yang kemudian ditransformasi menjadi filter digital maka hasil desain dengan metode ini akan memiliki bentuk dan karakteristik yang sama dengan filter analog. Jika kita mendesain dengan menggunakan filter analog Butterworth, maka hasil desain kita adalah filter digital IIR Butterworth, demikian juga dengan Chebyshev dan Elliptic.

Walaupun secara umum filter digital hasil desain ini akan memiliki respon frekuensi yang sama dengan filter analog, terdapat beberapa perbedaan antara keduanya. Perbedaan pertama adalah, filter analog memiliki respon frekuensi sampai $f = \infty$, sedangkan filter digital hanya sampai $f_s/2$.

Dengan kata lain, titik $f = \infty$ pada filter analog digeser (dikompres atau '*wrap*') ke titik $f_s/2$. Proses ini akan berakibat pada pergeseran titik cut-off, f_c. Titik f_c pada filter digital akan sedikit tergeser ke kiri, seperti ditunjukkan pada Gambar 16.3.



Gambar 16.3
Pergeseran frekuensi cut-off pada transformasi bilinier

Untuk mengatasi permasalahan pergeseran frekuensi cut-off di atas maka pada saat kita mendesain filter analog untuk tujuan ini, kita harus mendesain dengan frekuensi cut-off yang sedikit lebih besar dari yang kita inginkan. Dengan demikian pada saat proses transformasi bilinier ke digital, frekuensi cut-off ini akan bergeser kembali ke nilai cut-off yang kita inginkan. Misalnya, jika kita akan mendesain filter digital IIR dengan $f_c = 2000\ Hz$, maka pada saat kita memulai dengan mendesain filter analognya kita harus menggunakan

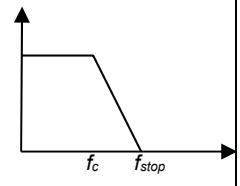
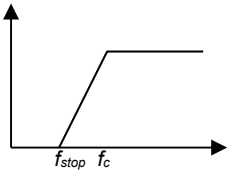
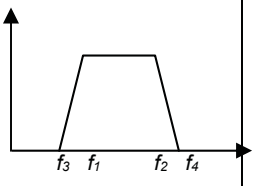
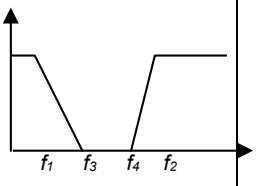
f_c = *2007 Hz*. Frekuensi *2007 Hz* ini disebut frekuensi *'unwrap'* dari *2000 Hz*.

Setelah mempelajari materi pada bab ini mahasiswa akan dapat mendesain filter IIR dengan metode transformasi bilinier, baik secara perhitungan manual maupun dengan menggunakan perangkat lunak pendukung.

16.1 Desain Filter IIR Butterworth

Langkah-langkah mendesain filter IIR Butterworth adalah sebagai berikut:

Langkah 1:
Tentukan spesifikasi filter digital:

LPF	HPF	BPF	BSF
δ_p, δ_s *(dalam dB)*, f_c , f_{stop}, f_s	δ_p, δ_s *(dalam dB)*, f_c , f_{stop}, f_s	δ_p, δ_s *(dalam dB)*, f_1 , f_2 , f_3 , f_4, f_s	δ_p, δ_s *(dalam dB)*, f_1 , f_2 , f_3 , f_4, f_s
			

Langkah 2:
Hitung spesifikasi filter digital ternormalisasi:

LPF	HPF	BPF	BSF
δ_p, δ_s ,	δ_p, δ_s ,	δ_p, δ_s ,	δ_p, δ_s ,
$\omega_{cn} = 2\pi (f_c/f_s)$	$\omega_{cn} = 2\pi (f_c/f_s)$	$\omega_{1n} = 2\pi (f_2/f_s)$	$\omega_{1n} = 2\pi (f_2/f_s)$
$\omega_{stop\ n} = 2\pi (f_{stop}/f_s)$	$\omega_{stop\ n} = 2\pi (f_{stop}/f_s)$	$\omega_{2n} = 2\pi (f_2/f_s)$	$\omega_{2n} = 2\pi (f_2/f_s)$
		$\omega_{3n} = 2\pi (f_3/f_s)$	$\omega_{3n} = 2\pi (f_3/f_s)$
		$\omega_{4n} = 2\pi (f_4/f_s)$	$\omega_{4n} = 2\pi (f_4/f_s)$
		$B = \omega_{2n} - \omega_{1n}$	$B = \omega_{2n} - \omega_{1n}$
		$\omega_{o\ n} = \sqrt{\omega_{1n}\omega_{2n}}$	$\omega_{o\ n} = \sqrt{\omega_{1n}\omega_{2n}}$

Langkah 3:

Unwrap frekuensi untuk mendapatkan frekuensi dari filter analog:

LPF	HPF	BPF	BSF
$\delta_p, \delta_s,$ $\omega_c = 2\tan\left(\frac{\omega_{cn}}{2}\right)$ $\omega_{stop} = 2\tan\left(\frac{\omega_{stop\,n}}{2}\right)$	$\delta_p, \delta_s,$ $\omega_c = 2\tan\left(\frac{\omega_{cn}}{2}\right)$ $\omega_{stop} = 2\tan\left(\frac{\omega_{stop\,n}}{2}\right)$	$\delta_p, \delta_s,$ $\omega_1 = 2\tan\left(\frac{\omega_{1n}}{2}\right)$ $\omega_2 = 2\tan\left(\frac{\omega_{2n}}{2}\right)$ $\omega_3 = 2\tan\left(\frac{\omega_{3n}}{2}\right)$ $\omega_4 = 2\tan\left(\frac{\omega_{4n}}{2}\right)$ $B = \omega_2 - \omega_1$ $\omega_o = \sqrt{\omega_1\omega_2}$	$\delta_p, \delta_s,$ $\omega_1 = 2\tan\left(\frac{\omega_{1n}}{2}\right)$ $\omega_2 = 2\tan\left(\frac{\omega_{2n}}{2}\right)$ $\omega_3 = 2\tan\left(\frac{\omega_{3n}}{2}\right)$ $\omega_4 = 2\tan\left(\frac{\omega_{4n}}{2}\right)$ $B = \omega_2 - \omega_1$ $\omega_o = \sqrt{\omega_1\omega_2}$

Langkah 4:

Filter analog didesain melalui prototype filter. Semua prototype filter analog dalam bentuk LPF dengan frekuensi cut-off pada 1 rad/s. Untuk itu, spesifikasi filter analog yang diperoleh di langkah ke 3 harus ditransfer ke bentuk dasar prototype (LPF 1 rad/s). Proses transfer adalah:

LPF	HPF	BPF	BSF
$\delta_p, \delta_s,$ $\Omega_c = 1$ $\Omega_{stop} = \frac{\omega_{stop}}{\omega_c}$	$\delta_p, \delta_s,$ $\Omega_c = 1$ $\Omega_{stop} = \frac{\omega_c}{\omega_{stop}}$	$\delta_p, \delta_s,$ $\Omega_c = 1$ $\Omega_{stop} = \frac{\omega_4 - \omega_3}{\omega_2 - \omega_1}$	$\delta_p, \delta_s,$ $\Omega_c = 1$ $\Omega_{stop} = \frac{\omega_2 - \omega_1}{\omega_4 - \omega_3}$

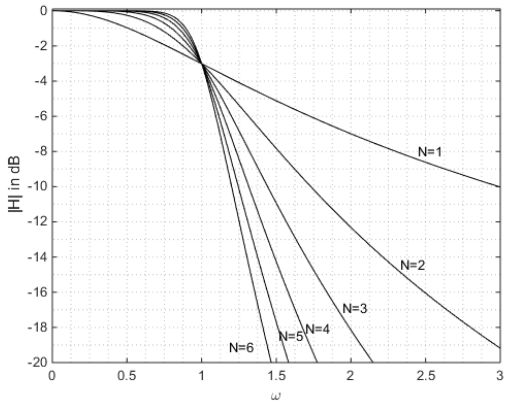
Langkah 5:

Merancang filter analog LPF dengan frekuensi cut-off sebesar 1 rad/s, dan frekuensi stopband Ω_{stop}. Filter seperti LPF dengan cut-off pada 1 rad/s disebut ***filter prototype***. Untuk filter Butterworth:

Butterworth

Orde filter (N):

Temukan letak titik (Ω_{stop}, δ_s) pada gambar di bawah ini kemudian tentukan N terkecil yang memiliki kecuraman transition band lebih curam atau sama dengan kecuraman yang dimaksud.



Cara lain menemukan N adalah dengan menggunakan rumus di bawah ini:

$$N = \frac{\log\left(\frac{10^{0.1\,\delta_{stop}} - 1}{\varepsilon^2}\right)}{\log\left(\Omega_{stop}^2\right)} \quad \text{dengan} \quad \varepsilon = \sqrt{10^{0.1\,\delta_p} - 1}$$

Tulis transfer function, H(s), dari filter LPF 1 rad/s (prototype) tersebut:

$$H(s) = \frac{1}{Den(s)}$$

N	Den (s)
1	(s + 1)
2	(s^2 + 1.414s + 1)
3	(s^2 + s + 1) (s + 1)
4	(s^2 + 0.76537s + 1) (s^2 + 1.8477s + 1)
5	(s^2 + 0.61803s + 1) (s^2 + 1.61803s + 1) (s + 1)

Tabel filter prototpe di atas dapat diperoleh dengan perintah:

```
>> [z,p,k] = buttap(n);
>> [num,den]=zp2tf(z,p,k)
```

Hasil perhitungan tersimpan dalam variable 'num' (koefisien dari numerator) dan 'den' (koefisien dari denumerator). Sebagai contoh, jika kita akan mencari prototype dari Butterworth dengan n = 3 maka kita gunakan perintah:

```
>> [z,p,k] = buttap(3);
>> [num,den]=zp2tf(z,p,k)

num =
         0         0         0    1.0000

den =
    1.0000    2.0000    2.0000    1.0000
```

Hasil ini berarti:

$$H(s) = \frac{0s^3 + 0s^2 + 0s + 1}{s^3 + 2s^2 + 2s + 1} = \frac{1}{s^3 + 2s^2 + 2s + 1}$$

Langkah 6:

Transformasikan kembali prototype LPF 1 rad/s tersebut ke filter analog yang seharusnya (LPF atau HPF atau BPF atau BSF) dengan frekuensi sesuai dengan ω_c, ω_{stop}, ω_1, ω_2, ω_3, dan ω_4.

LPF	HPF	BPF	BSF
Gantilah s dengan $\left(\frac{1}{\omega_c}\right)s$	*Gantilah s dengan* $\left(\frac{\omega_c}{s}\right)$	*Gantilah s dengan* $\left(\frac{s^2+\omega_0^2}{Bs}\right)$	*Gantilah s dengan* $\left(\frac{Bs}{s^2+\omega_0^2}\right)$

Setelah langkah ini maka kita telah memperoleh filter analog sesuai dengan spesifikasi pada langkah ke-3

Langkah 7:
Transformasikan filter analog tersebut ke filter digital IIR dengan menggunakan transformasi bilinier.

Transformasi Bilinier
Gantilah *s* dengan $$s = \frac{2(1 - z^{-1})}{(1 + z^{-1})}$$

16.2 Contoh Desain Filter IIR Butterworth

Sebagai contoh, kita akan mendesain suatu filter IIR Butterworth dengan spesifikasi sebagai berikut:

Langkah 1:

Tipe filter	: LPF, Butterworth
Passband ripple, δ_p	: 3 dB
Stopband attenuation, δ_{stop}	: 10 dB
Frekuensi cut-off, f_c	: 2 kHz
Frekuensi stoband, f_{stop}	: 3 kHz
Frekuensi sampling, f_s	: 10 kHz

Langkah 2: spesifikasi filter ternormalisasi

Frekuensi cut-off, $\omega_{cn} = 2\pi\left(\frac{2\,kHz}{10\,kHz}\right) = 1.2566$ rad/s

Frekuensi stopband, $\omega_{stop\,n} = 2\pi\left(\frac{3\,kHz}{10\,kHz}\right) = 1.8850$ rad/s

Langkah 3: frekuensi unwrap

Frekuensi cut-off, $\omega_c = 2\tan\left(\frac{1.2566}{2}\right) = 1.4531\ rad/s$

Frekuensi stopband, $\omega_{stop} = 2\tan\left(\frac{1.8850}{2}\right) = 2.7528\ rad/s$

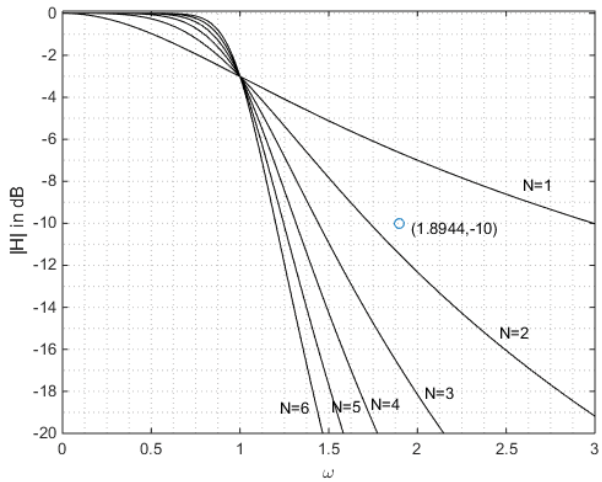
Langkah 4: frekuensi prototype

Frekuensi cut-off, $\Omega_c = 1$

Frekuensi stopband, $\Omega_{stop} = \frac{\omega_{stop}}{\omega_c} = \frac{2.7528}{1.4531} = 1.8944$

Langkah 5: orde filter

Kita dapat menentukan orde filter dengan menggunakan grafik Butterworth. Kita tempatkan titik (Ω_{stop}, δ_s) = (1.8944, -10dB) pada kurva Butterworth seperti Gambar 16.4 di bawah ini.



Gambar 16.4
Menentukan orde filter dengan grafik

Titik tersebut berada antaran N = 1 dan N = 2 sehingga kita memilih yang lebih besar yatu N = 2.

Cara lain adalah dengan menggunakan rumus:

$$\varepsilon = \sqrt{10^{0.1\,\delta_p} - 1} = \sqrt{10^{0.1\,(3)} - 1} = 0.9976$$

$$N = \frac{\log\left(\frac{10^{0.1\,\delta_{stop}} - 1}{\varepsilon^2}\right)}{\log\left(\Omega_{stop}^2\right)} = \frac{\log\left(\frac{10^{0.1(10)} - 1}{(0.9976)^2}\right)}{\log\left(1.8944^2\right)} = 1.7233$$

jadi kita menggunakan prototype filter dengan orde bilangan bulat lebih besar dari 1.7233 sehingga kita memilih N = 2, maka

$$H(s) = \frac{1}{s^2 + 1.414s + 1}$$

Langkah 6:
transformasi prototype tersebut ke filter LPF dengan $\omega_c = 1.4531\ rad/s$

Cara transformasi adalah dengan mengganti semua *s* dengan

$$\left(\frac{1}{\omega_c}\right)s = \left(\frac{1}{1.4531}\right)s = 0.6882s$$

$$H(s) = \frac{1}{(0.6882s)^2 + 1.414(0.6882s) + 1}$$

$$H(s) = \frac{1}{0.4736s^2 + 0.9731s + 1}$$

Langkah 7: transformasi bilinier,

dengan mengganti semua s dengan $\frac{2(1-z^{-1})}{(1+z^{-1})}$

$$H(z)=\frac{1}{0.4736\left(\frac{2(1-z^{-1})}{1+z^{-1}}\right)^2+0.9731\left(\frac{2(1-z^{-1})}{1+z^{-1}}\right)+1}$$

$$H(z)=\frac{1}{\frac{1.8944(1-z^{-1})^2}{(1+z^{-1})^2}+\frac{1.9462(1-z^{-1})(1+z^{-1})}{(1+z^{-1})^2}+\frac{(1+z^{-1})^2}{(1+z^{-1})^2}}$$

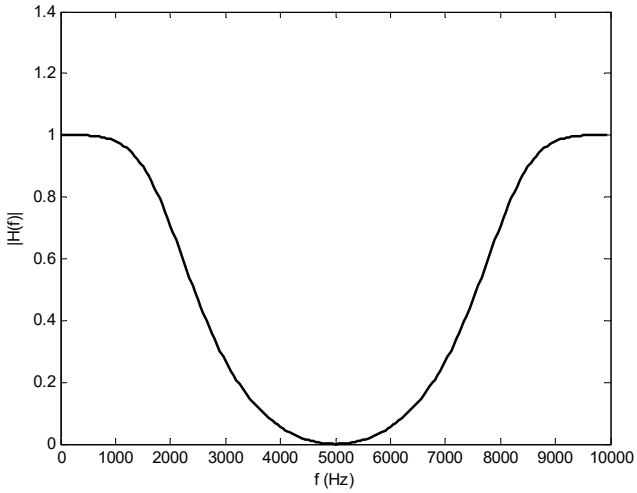
$$H(z)=\frac{(1+z^{-1})^2}{1.8944(1-z^{-1})^2+1.9462(1-z^{-1})(1+z^{-1})+(1+z^{-1})^2}$$

$$H(z)=\frac{(1+2z^{-1}+z^{-2})}{1.8944(1-2z^{-1}+z^{-2})+1.9462(1-z^{-2})+(1+2z^{-1}+z^{-2})}$$

$$H(z)=\frac{(1+2z^{-1}+z^{-2})}{4.8406-1.7888z^{-1}+0.9482z^{-2}}$$

$$H(z)=\frac{0.2066+0.4132z^{-1}+0.2066z^{-2}}{1-0.3695z^{-1}+0.1959z^{-2}}$$

Respon frekuensi dari filter tersebut adalah seperti pada Gambar 16.5. Gambar tersebut mengkonformasi bahwa hasil desain ini adalah sesuai spesifikasi filter yang diminta.



Gambar 16.5
Respon frekuensi dari filter LPF Butterworth hasil desain

16.3 Desain Filter IIR Butterworth dengan Matlab

Jika kita ingin mengunakan Matlab untuk melakukan proses desain dari contoh tersebut maka kita menggunakan perintah:

```
>> N=2;
>> fc=2000/5000;
>> [B,A]=butter(N,fc,'low');
```

Perintah tersebut menghasilkan

```
B =
   0.2066    0.4131    0.2066

A =
   1.0000   -0.3695    0.1958
```

artinya

$$H(z) = \frac{0.2066 + 0.4131z^{-1} + 0.2066z^{-2}}{1 - 0.3695z^{-1} + 0.1958z^{-2}}$$

Hasil ini persis sama dengan hasil ang diperoleh dari perhitungan secara manual di atas.

Untuk menggambarkan respon frekuensi dari filter hasil desain tersebut seperti pada Gambar 16.5 maka gunakan perintah sebagai berikut:

```
>> B =[0.2066    0.4131    0.2066];
>> A =[1.0000   -0.3695    0.1958];
>> [H,W] = freqz(B,A,200,1,'whole');
>> plot(W*10000,abs(H),'Color',[0 0 0],'LineWidth',2)
>> xlabel('f (Hz)');
>> ylabel('|H(f)|');
```

16.4 Desain Filter IIR Chebyshev Tipe I

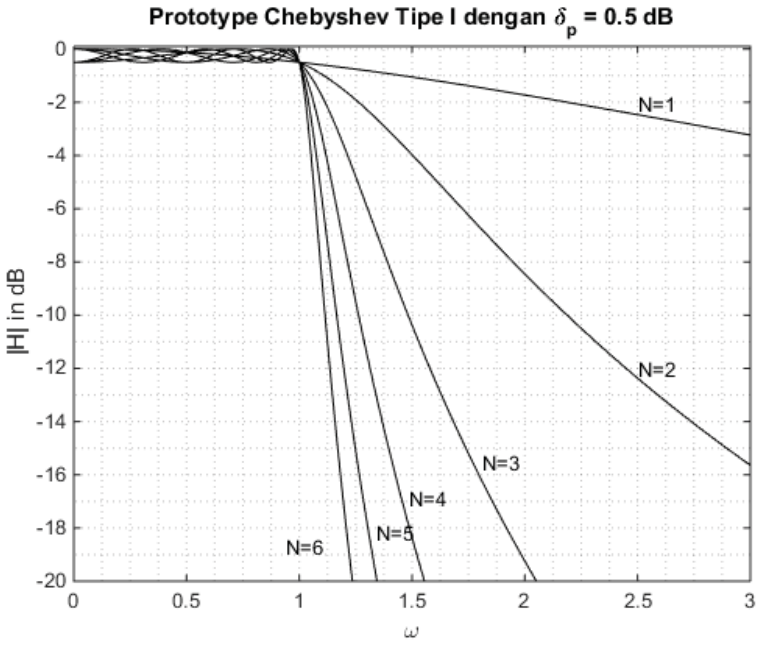
Langkah-langkah mendesain filter IIR Chebyshev Tipe I hampir sama seperti langkah desain filter Butterworth di atas. Perbedaannya terletak pada langkah 5.

Langkah 5:

Merancang filter analog LPF dengan frekuensi cut-off sebesar 1 rad/s, dan frekuensi stopband Ω_{stop}. Untuk filter Chebyshev Tipe I:

Chebyshev Tipe I

Orde filter :



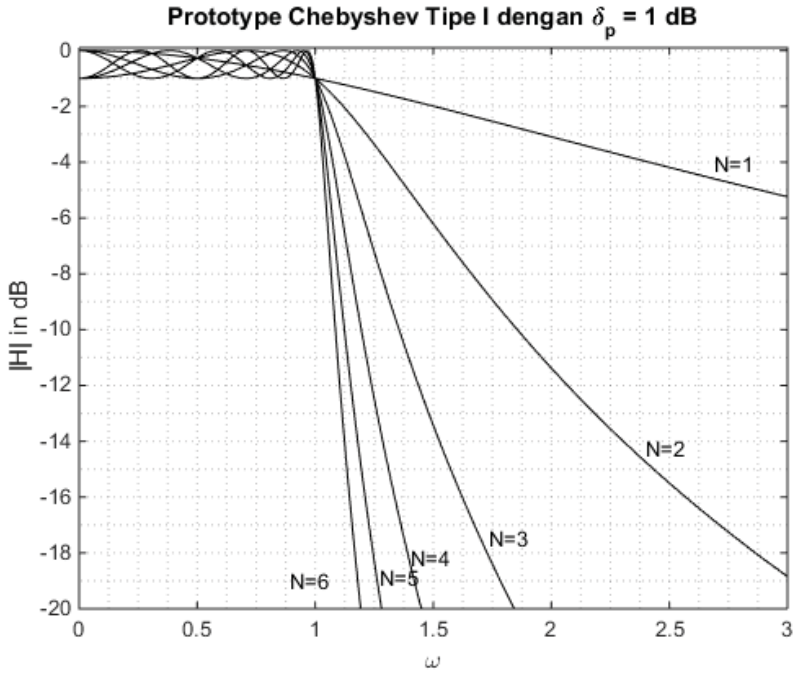
$$N = \frac{\cosh^{-1}\left(\sqrt{\frac{10^{0.1\,\delta_{stop}} - 1}{\varepsilon^2}}\right)}{\cosh^{-1}(\Omega_{stop})} \quad \text{dengan} \quad \varepsilon = \sqrt{10^{0.1\,\delta_p} - 1}$$

Tulis transfer function, H(s), dari filter LPF 1 rad/s tersebut:

Chebyshev Tipe I (prototype) 1 rad/s (passband ripple = 0.5 dB atau ε = 0.3493)

N	H(s)
1	$\frac{2.8628}{s + 2.8628}$
2	$\frac{1.4314}{s^2 + 1.4256s + 1.5162}$
3	$\frac{0.7157}{s^3 + 1.2529s^2 + 1.5349s + 0.7157}$
4	$\frac{0.3579}{s^4 + 1.1974s^3 + 1.7169s^2 + 1.0255s + 0.3791}$
5	$\frac{0.1789}{s^5 + 1.1725s^4 + 1.9347s^3 + 1.3096s^2 + 0.7525s + 0.1789}$

Untuk Chebyshev Tipe I dengan passband ripple 1 dB:



Chebyshev Tipe I (prototype) 1 rad/s (passband ripple = 1.0 dB atau ε = 0.5088)

N	H(s)
1	$\frac{1.9652}{s+1.9652}$
2	$\frac{0.9826}{s^2+1.0977s+1.1025}$
3	$\frac{0.4913}{s^3+0.9883s^2+1.2384s+0.4913}$
4	$\frac{0.2456}{s^4+0.9528s^3+1.4539s^2+0.7426s+0.2756}$
5	$\frac{0.0614}{s^5+0.9368s^4+1.6888s^3+0.9744s^2+0.5805s+0.0614}$

Tabel filter prototype di atas dapat diperoleh dengan perintah:

```
>> [z,p,k] = cheb1ap(n,ripple);
>> [num,den]=zp2tf(z,p,k)
```

Hasil perhitungan tersimpan dalam variable 'num' (koefisien dari numerator) dan 'den' (koefisien dari denumerator). Sebagai contoh, jika kita akan mencari protorype dari Chebyshev tipe I dengan n = 3 dan $\delta_p = 1dB$ maka kita gunakan perintah:

```
>> [z,p,k] = cheb1ap(3,1);
>> [num,den]=zp2tf(z,p,k)
num =
         0         0         0    0.4913

den =
    1.0000    0.9883    1.2384    0.4913
```

Hasil ini berarti:

$$H(s) = \frac{0s^3 + 0s^2 + 0s + 0.4913}{s^3 + 0.9883s^2 + 1.2384s + 0.4913} = \frac{0.4913}{s^3 + 0.9883s^2 + 1.2384s + 0.4913}$$

16.5 Contoh Desain Filter IIR Chebyshev Tipe I

Sebagai contoh, kita akan mendesain suatu filter IIR Chebyshev tipe I dengan spesifikasi sebagai berikut:

Langkah 1:

Tipe filter	: HPF, Chebyshev Tipe I
Passband ripple, δ_p	: 1 dB
Stopband attenuation	: 10 dB
Frekuensi cut-off, f_c	: 3 kHz

Frekuensi stoband, f_{stop} : 2 kHz
Frekuensi sampling, f_s : 10 kHz

Langkah 2: spesifikasi filter ternormalisasi

Frekuensi cut-off, $\omega_{cn} = 2\pi\left(\frac{3\,kHz}{10\,kHz}\right) = 1.8850$ rad/s

Frekuensi stopband, $\omega_{stopn} = 2\pi\left(\frac{2\,kHz}{10\,kHz}\right) = 1.2566$ rad/s

Langkah 3: frekuensi unwrap

Frekuensi cut-off, $\omega_c = 2\tan\left(\frac{1.8850}{2}\right) = 2.7528\ rad/s$

Frekuensi stopband, $\omega_{stop} = 2\tan\left(\frac{1.2566}{2}\right) = 1.4531\ rad/s$

Langkah 4: frekuensi prototype

Frekuensi cut, $\Omega_c = 1$

Frekuensi stopband, $\Omega_{stop} = \frac{\omega_c}{\omega_{stop}} = \frac{2.7528}{1.4513} = 1.8944$

Langkah 5: orde filter

$$\varepsilon = \sqrt{10^{0.1\,\delta_p} - 1} = \sqrt{10^{0.1\,(3)} - 1} = 0.5088$$

$$N = \frac{\cosh^{-1}\left(\sqrt{\frac{10^{0.1\,\delta_{stop}} - 1}{\varepsilon^2}}\right)}{\cosh^{-1}(\Omega_{stop})} = \frac{\cosh^{-1}\left(\sqrt{\frac{10^{0.1\,(10)} - 1}{0.5088^2}}\right)}{\cosh^{-1}(1.8944)} = 1.96$$

jadi kita menggunakan prototype filter dengan orde 2, maka

$$H(s) = \frac{0.9826}{s^2 + 1.0977s + 1.1025}$$

Langkah 6: transformasi prototype tersebut ke filter HPF dengan

$\omega_c = 2.7528\ rad/s$ dengan cara

mengganti semua *s* dengan $\left(\frac{\omega_c}{s}\right) = \left(\frac{2.7528}{s}\right)$

$$H(s) = \frac{0.9826}{\left(\frac{2.7528}{s}\right)^2 + 1.0977\left(\frac{2.7528}{s}\right) + 1.1025}$$

$$H(s) = \frac{0.9826}{\left(\frac{7.5779}{s^2}\right) + \left(\frac{3.0217}{s}\right) + 1.1025}$$

$$H(s) = \frac{0.9826s^2}{1.1025s^2 + 3.0217s + 7.5779}$$

Langkah 7: transformasi bilinier, dengan mengganti semua s dengan $\frac{2(1-z^{-1})}{(1+z^{-1})}$

$$H(z) = \frac{0.9826\left(\frac{2(1-z^{-1})}{1+z^{-1}}\right)^2}{1.1025\left(\frac{2(1-z^{-1})}{1+z^{-1}}\right)^2 + 3.0217\left(\frac{2(1-z^{-1})}{1+z^{-1}}\right) + 7.5779}$$

$$H(z) = \frac{\dfrac{3.9304(1-z^{-1})^2}{(1+z^{-1})^2}}{\dfrac{4.4100(1-z^{-1})^2}{(1+z^{-1})^2} + \dfrac{6.0434(1-z^{-1})(1+z^{-1})}{(1+z^{-1})^2} + \dfrac{7.5779(1+z^{-1})^2}{(1+z^{-1})^2}}$$

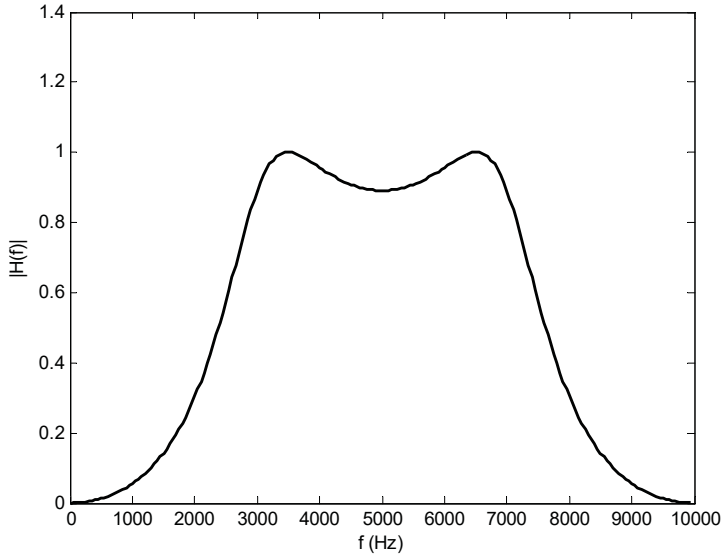
$$H(z) = \frac{3.9304(1-z^{-1})^2}{4.4100(1-z^{-1})^2 + 6.0434(1-z^{-1})(1+z^{-1}) + 7.5779(1+z^{-1})^2}$$

$$H(z) = \frac{(3.9304 - 7.8608z^{-1} + 3.9304z^{-2})}{(4.4100 - 8.8200z^{-1} + 4.4100z^{-2}) + (6.0434 - 6.0434z^{-2}) + (7.5779 + 15.1558z^{-1} + 7.5779z^{-2})}$$

$$H(z) = \frac{(3.9304 - 7.8608z^{-1} + 3.9304z^{-2})}{(18.0313 + 6.3358z^{-1} + 5.9445z^{-2})}$$

$$H(z) = \frac{(0.2180 - 0.4358z^{-1} + 0.2180z^{-2})}{(1 + 0.3514z^{-1} + 0.3297z^{-2})}$$

Respon frekuensi dari filter tersebut adalah seperti pada Gambar 16.6.



Gambar 16.6
Respon frekuensi dari filter HPF Chebyshev Tipe I hasil desain

16.6 Desain Filter IIR Chebyshev Tipe I dengan Matlab

Contoh perintah Matlab untuk filter LPF Chebyshev Tipe I orde 2 dengan $f_c = 2\ kHz$, $f_s = 10\ kHz$ dan $\delta_p = 0.1dB$ adalah:

```
>> N=2;
>> r=0.1;
>> fc=2000/5000;
>> [B,A]=cheby1(N,r,fc,'low')

B =
   0.3866    0.7732    0.3866
A =
   1.0000    0.3351    0.2293
```

artinya

$$H(z) = \frac{0.3866 + 0.7732z^{-1} + 0.3866z^{-2}}{1 + 0.3351z^{-1} + 0.2293z^{-2}}$$

Untuk menggambarkan respon frekuensi dari filter hasil desain tersebut gunakan:

```
>> B =[0.3866    0.7732    0.3866];
>> A =[1.0000    0.3351    0.2293];
>> [H,W] = freqz(B,A,200,1,'whole');
>> plot(W*10000,abs(H),'Color',[0 0 0],'LineWidth',2)
>> xlabel('f (Hz)');
>> ylabel('|H(f)|');
```

16.7 Desain Filter IIR Chebyshev Tipe II

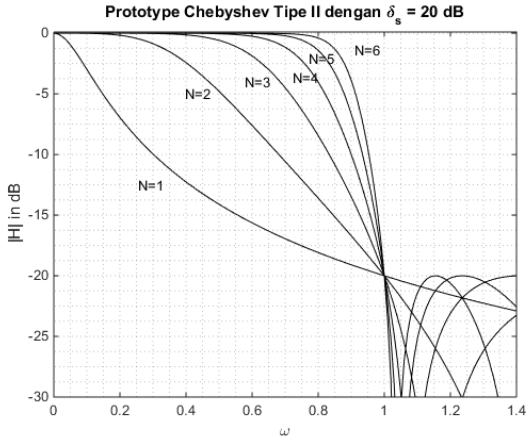
Langkah-langkah mendesain filter IIR Chebyshev Tipe II sama seperti langkah desain filter Chebyshev tipe I di atas. Perbedaannya terletak pada langkah 5.

Langkah 5:

Merancang filter analog LPF dengan frekuensi cut-off sebesar 1 rad/s, dan frekuensi stopband Ω_{stop}. Untuk filter Chebyshev Tipe II:

Chebyshev Tipe II

Orde filter :



$$N = \frac{\cosh^{-1}\left(\sqrt{\frac{10^{0.1\,\delta_{stop}} - 1}{\varepsilon^2}}\right)}{\cosh^{-1}(\Omega_{stop})} \quad \text{dengan} \quad \varepsilon = \sqrt{10^{0.1\,\delta_p} - 1}$$

Tulis transfer function, H(s), dari filter LPF 1 rad/s tersebut:

Chebyshev Tipe II (prototype) 1 rad/s (passband attenuation, δ_{stop} = 20 dB)

N	H(s)
1	$\frac{0.1005}{s + 0.1005}$
2	$\frac{s^2 + 0.2}{s^2 + 0.6s + 0.2}$
3	$\frac{0.3015s^2 + 0.4020}{s^3 + 1.4054s^2 + 0.9421s + 0.4020}$
4	$\frac{s^4 + 0.8s^2 + 0.8}{s^4 + 2.2615^3 + 2.6371s^2 + 1.7145s + 0.8}$
5	$\frac{0.5025s^4 + 2.0101s^2 + 1.6081}{s^5 + 3.2471s^4 + 5.1455s^3 + 5.4057s^2 + 3.3047s + 1.6081}$

Tabel filter prototype di atas dapat diperoleh dengan perintah:

```
>> [z,p,k] = cheb2ap(n,stopband attenuation);
```

```
>> [num,den]=zp2tf(z,p,k)
```

Hasil perhitungan tersimpan dalam variable 'num' (koefisien dari numerator) dan 'den' (koefisien dari denumerator). Sebagai contoh, jika kita akan mencari protorype dari Chebyshev tipe II dengan n = 3 dan $\delta_{stop} = 20dB$ maka kita gunakan perintah:

```
>> [z,p,k] = cheb2ap(3,20);
>> [num,den]=zp2tf(z,p,k)
num =
        0    0.3015         0    0.4020

den =
   1.0000    1.4054    0.9421    0.4020
```

Hasil ini berarti:

$$H(s) = \frac{0s^3 + 0.3015s^2 + 0s + 0.4020}{s^3 + 1.4054s^2 + 0.9421s + 0.4020}$$
$$= \frac{0.3015s^2 + 0.4020}{s^3 + 1.4054s^2 + 0.9421s + 0.4020}$$

16.8 Desain Filter IIR Chebyshev Tipe II dengan Matlab

Contoh perintah Matlab untuk filter LPF Chebyshev Tipe II orde 2 dengan $f_c = 2\ kHz$, $f_s = 10\ kHz$ dan $\delta_{stop} = 20dB$ adalah:

```
>> N=2;
>> r=20;
>> fc=2000/5000;
>> [B,A]=cheby2(N,r,fc,'low')

B =
```

```
    0.1334    0.0072    0.1334

A =
    1.0000   -1.1605    0.4344
```

artinya

$$H(z) = \frac{0.1334 + 0.0072z^{-1} + 0.1334z^{-2}}{1 - 1.1605z^{-1} + 0.4344z^{-2}}$$

Untuk menggambarkan respon frekuensi dari filter hasil desain tersebut gunakan:

```
>> B =[0.1334    0.0072    0.1334];
>> A =[1.0000   -1.1605    0.4344];
>> [H,W] = freqz(B,A,200,1,'whole');
>> plot(W*10000,abs(H),'Color',[0 0 0],'LineWidth',2)
>> xlabel('f (Hz)');
>> ylabel('|H(f)|');
```

16.9 Desain Filter IIR Elliptic

Langkah-langkah mendesain filter IIR Elliptic sama seperti langkah desain filter-filter di atas. Perbedaannya terletak pada langkah 5.

Langkah 5:

Merancang filter analog LPF dengan frekuensi cut-off sebesar 1 rad/s, dan frekuensi stopband Ω_{stop}. Untuk filter Elliptic:

Elliptic

Orde filter :

$$N = \frac{CEI(rt)\ CEI\left(\sqrt{1-kn^2}\right)}{CEI\left(\sqrt{1-rt^2}\right)CEI(kn)} \quad \text{dengan} \quad kn = \sqrt{\frac{10^{0.1\,\delta_p}-1}{10^{0.1\,\delta_{stop}}-1}}$$

$$rt = \frac{\Omega c}{\Omega_{stop}} \text{ dan } CEI(k) = \int_0^{2\pi} \sqrt{1-k^2\sin^2 x}\ dx$$

Tulis transfer function, H(s), dari filter LPF 1 rad/s tersebut:

Elliptic (prototype) 1 rad/s (passband ripple = 0.5 dB dan stopband attenuation = 20 dB)

N	H(s)
1	$\frac{2.8628}{s+2.8628}$
2	$\frac{s^2+1.4742}{s^2+1.3430s^2+1.5616}$
3	$\frac{0.3642s^2+0.9094}{s^3+1.2197s^2+1.4870s+0.9094}$
4	$\frac{s^4+0.6336s^2+0.6837}{s^4+1.1351s^3+1.8831s^2+1.1387s+0.7242}$
5	$\frac{0.3099s^4+0.8813s^2+0.5983}{s^5+1.1527s^4+2.2236s^3+1.7673s^2+1.2003s+0.5983}$

Tabel filter prototype di atas dapat diperoleh dengan perintah:

```
>> [z,p,k] = ellipap(n,ripple passband,ripple stopband);
>> [num,den]=zp2tf(z,p,k)
```

Hasil perhitungan tersimpan dalam variable ‘num’ (koefisien dari numerator) dan ‘den’ (koefisien dari denumerator). Sebagai contoh, jika kita akan mencari protorype dari Elliptic

dengan n = 3 dan $\delta_p = 0.5dB$ $\delta_{stop} = 20dB$ maka kita gunakan perintah:

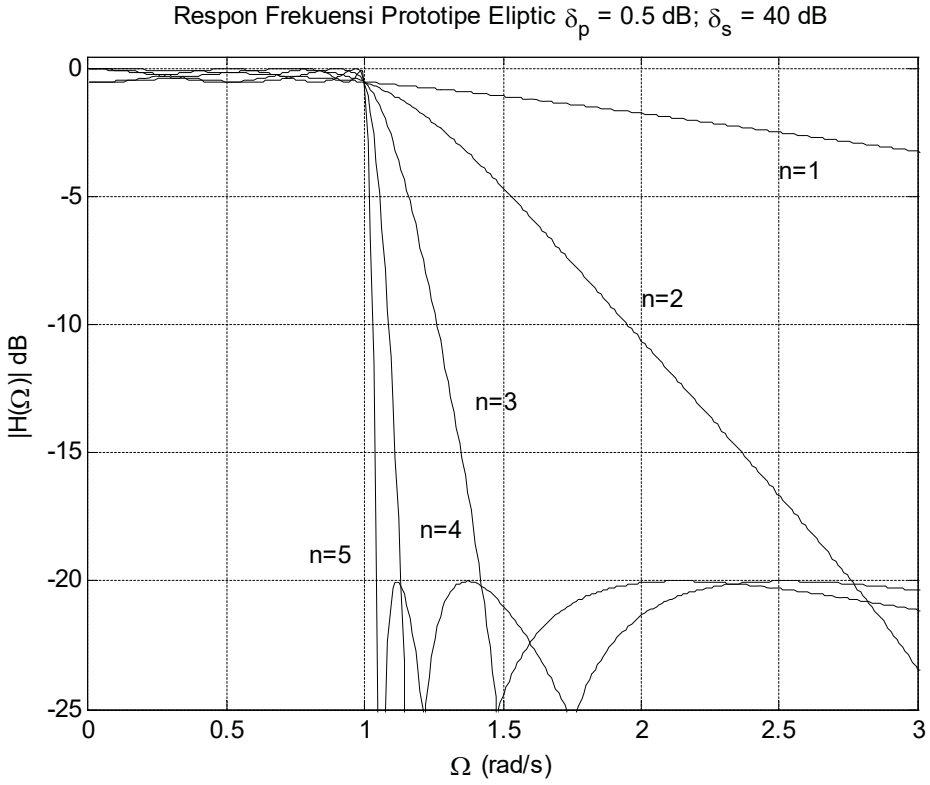
```
>> [z,p,k] = ellipap(3,0.5,20);
>> [num,den]=zp2tf(z,p,k)
num =
         0    0.3642         0    0.9094

den =
    1.0000    1.2197    1.4870    0.9094
```

Hasil ini berarti:

$$H(s) = \frac{0.3642s^2 + 0.9094}{s^3 + 1.2197s^2 + 1.4870s + 0.9094}$$

16.10 Menentukan orde filter Elliptic

Orde dari filter Elliptic ditentukan dengan menggunakan rumus di dalam kotak di atas. Penentuan orde filter ini menjadi sangat rumit karena rumus di atas menggunakan integral yang banyak. Untuk memudahkan proses di atas maka digunakan metode grafik. Penentuan orde filter dilakukan dengan menggambarkan titik (Ω_{stop} , δ_{stop}) pada Gambar 16.7. Jika titik tersebut berada diantara garis dari dua buah orde, maka yang digunakan adalah orde terbesar dari kedua orde tersebut.



Gambar 16.7
Respon Frekuensi Protoripe Elliptic δ_p=0.5 dB dan δ_{stop}=20 dB

Sebagai contoh, jika filter yang akan dirancang memiliki $\Omega_{stop} = 1.89$ dan $\delta_{stop} = 20dB$, maka titik tersebut akan terletak di antara $N = 2$ dan $N = 3$. Dengan demikian maka filter yang dirancang harus memiliki orde, N, sebesar 3.

16.11 Desain Filter IIR Elliptic dengan Matlab

Contoh perintah Matlab untuk filter LPF Elliptic orde 2 dengan $f_c = 2$ kHz, $f_s = 10$ kHz dan $\delta_p = 0.5dB$, $\delta_{stop} = 20dB$ adalah:

```
>> N=2;
>> rp=0.5;
>> rs=20;
>> fc=2000/5000;
>> [B,A]=ellip(N,rp,rs,fc,'low')

B =
   0.3136    0.4844    0.3136

A =
   1.0000   -0.1255    0.3031
```

artinya

$$H(z) = \frac{0.3136 + 0.4844z^{-1} + 0.3136z^{-2}}{1 - 0.1255z^{-1} + 0.3031z^{-2}}$$

Untuk menggambarkan respon frekuensi dari filter hasil desain tersebut gunakan:

```
>> B =[0.3136    0.4844    0.3136];
>> A =[1.0000   -0.1255    0.3031];
>> [H,W] = freqz(B,A,200,1,'whole');
>> plot(W*10000,abs(H),'Color',[0 0 0],'LineWidth',2)
>> xlabel('f (Hz)');
>> ylabel('|H(f)|');
```

SOAL LATIHAN

1. Dengan metode transformasi bilinier, rancanglah suatu filter
   Tipe filter : HPF, Butterworth
   Passband ripple, δ_p : 3 dB
   Stopband attenuation : 10 dB
   Frekuensi cut-off, f_c : 2 kHz
   Frekuensi stoband, f_{stop} : 1 kHz
   Frekuensi sampling, f_s : 10 kHz

2. Amati respon frekuensi dari filter yang dirancang pada soal nomor 1 di atas. Apakah sesuai dengan spesifikasi rancangan?

3. Dengan metode transformasi bilinier, rancanglah suatu filter
   Tipe filter : LPF, Chebychev Type 1
   Passband ripple, δ_p : 0.1 dB
   Orde : 2
   Frekuensi cut-off, f_c : 2 kHz
   Frekuensi stoband, f_{stop} : 1 kHz
   Frekuensi sampling, f_s : 10 kHz

4. Amati respon frekuensi dari filter yang dirancang pada soal nomor 3 di atas. Apakah sesuai dengan spesifikasi rancangan?

5. Dengan metode transformasi bilinier, rancanglah suatu filter
   Tipe filter : LPF, Chebychev Type II
   Stopband attenuation, δ_s : 20 dB
   Orde : 2
   Frekuensi cut-off, f_c : 2 kHz
   Frekuensi stoband, f_{stop} : 1 kHz
   Frekuensi sampling, f_s : 10 kHz

6. Amati respon frekuensi dari filter yang dirancang pada soal nomor 5 di atas. Apakah sesuai dengan spesifikasi rancangan?

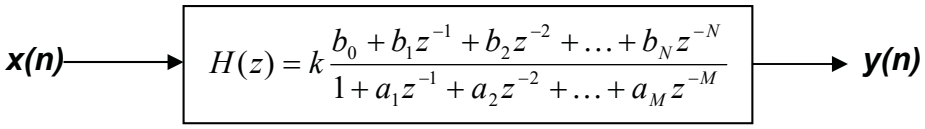
7. Dengan metode transformasi bilinier, rancanglah suatu filter
   Tipe filter : HPF, Eliptic

Passband ripple, δ_p	: 0.5 dB
Stopband attenuation	: 20 dB
Frekuensi cut-off, f_c	: 2 kHz
Frekuensi stoband, f_{stop}	: 1 kHz
Frekuensi sampling, f_s	: 10 kHz

8. Amati respon frekuensi dari filter yang dirancang pada soal nomor 7 di atas. Apakah sesuai dengan spesifikasi rancangan?

17
Desain Filter IIR dengan Metode Impulse Invariant

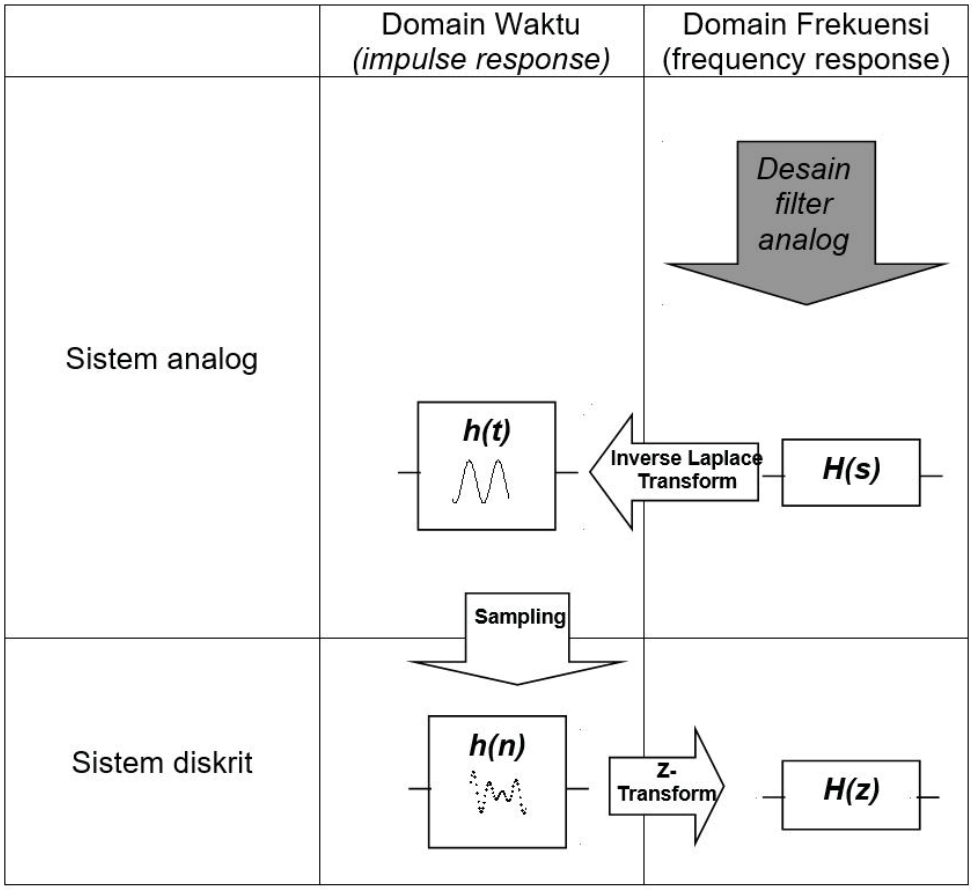
Pada bab ini kita akan mempelajari cara mendesain filter IIR dengan metode impulse invariant. Filter IIR biasanya dinyatakan dalam bentuk transfer function, *H(z)*, seperti pada Gambar 17.1. Hasil akhir dari proses desain suatu filter IIR dengan metode impulse invariant adalah nilai dari koefisien filter b_n, a_n, dan gain, k.



Gambar 17.1
Filter IIR dalam bentuk transfer function

Sama seperti pada metode transformasi bilinier, metode impulse invariant ini juga membuat filter digital berdasarkan bentuk dasar (transfer function) filter analog, *H(s)*. Oleh karena itu, proses desain filter IIR dengan metode ini dimulai dengan mendesain filter analog seperti pada metode

transformasi bilinier (Bab 16). Setelah menemukan transfer function, *H(s)*, dari filter analog maka langkah selanjutnya adalah menggunakan transformasi inverse Laplace untuk menemukan impulse response, *h(t)*, dari filter analog tersebut. Selanjutnya adalah mengambil sample dari impulse response tersebut, *h(n)*, yang kemudian ditransformasi dengan transformasi-Z untuk menjadi H(z). Gambar 17.2 menunjukkan proses desain filter FIR dengan metode impulse invariant.



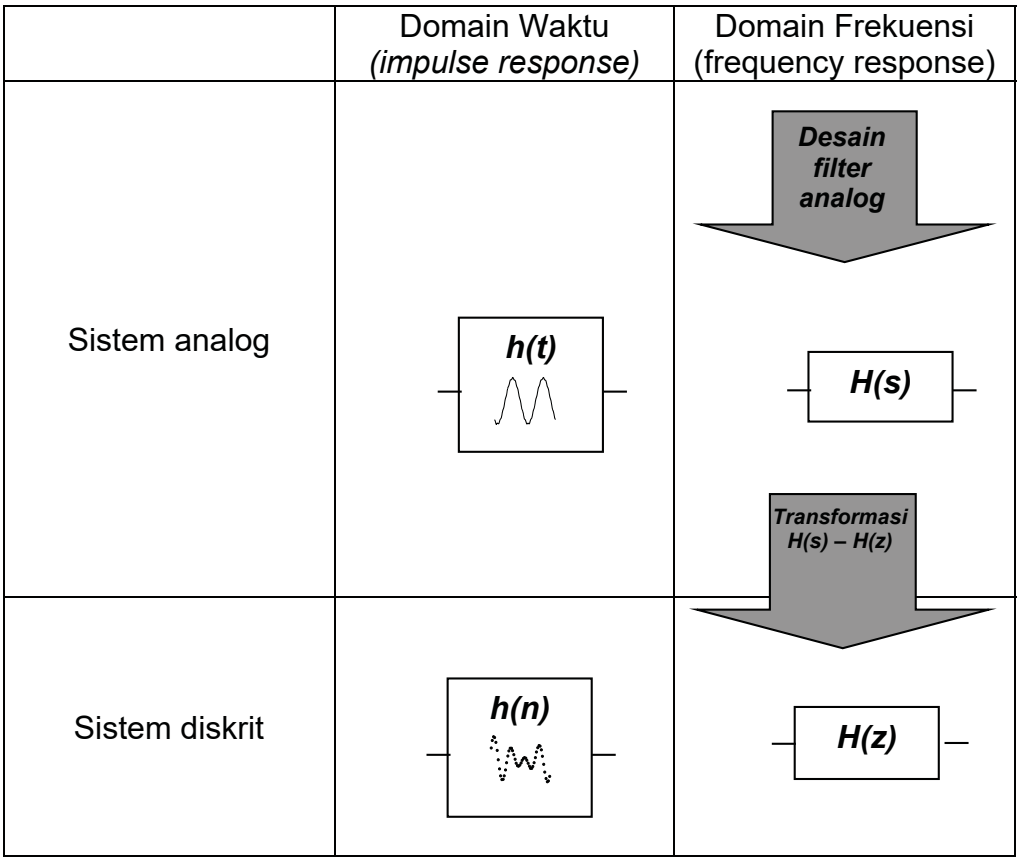
Gambar 17.2
Proses Desain Filter IIR dengan Metode Impulse Invariant

Karena proses desainnya dimulai dengan filter analog yang kemudian ditransformasi menjadi filter digital maka hasil desain dengan metode ini akan memiliki bentuk dan karakteristik yang sama dengan filter analog. Jika kita mendesain dengan menggunakan filter analog Butterworth, maka hasil desain kita adalah filter digital IIR Butterworth, demikian juga dengan Chebyshev dan elliptic. Metode desain ini akan menghasilkan filter digital dengan respon frekuensi yang lebih mirip filter analog jika dibandingkan dengan metode desain dengan transformasi bilinier. Proses ini tidak menyebabkan terjadinya peristiwa *wraping* frekuensi seperti pada metode desain dengan transformasi bilinier, sehingga frekuensi cut-off dari *H(z)* akan tepat sama dengan frekuensi cut-off dari *H(s)*.

Setelah mempelajari materi pada bab ini mahasiswa akan dapat mendesain filter IIR dengan metode *impulse invariant*, baik secara perhitungan manual maupun dengan menggunakan perangkat lunak pendukung.

17.1 Langkah Desain Filter IIR dengan Impulse Invariant

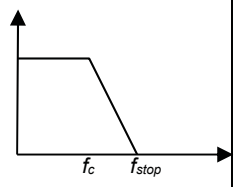
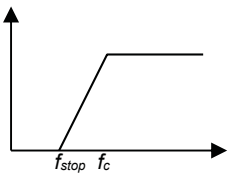
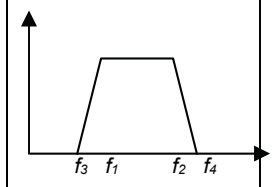
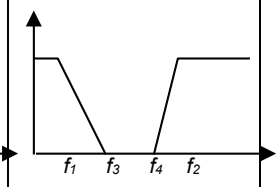
Proses desain filter IIR dengan metode impulse invariant seperti pada Gambar 17.2 terlihat sangat rumit dan matematis. Secara praktis, proses desain ini dipersingkat dengan menggunakan bentuk-bentuk umum sehingga dapat dilakukan dengan transformasi langsung dari *H(s)* ke *H(z)* seperti terlihat pada Gambar 17.3.



Gambar 17.3
Proses Desain Filter IIR dengan Metode Impulse Invariant secara praktis

Langkah-langkah mendesain filter IIR dengan metode impulse invariant dimulai dengan langkah-langkah untuk mendesain filter analog. Langkah-langkah mendesain filter analog hampir sama seperti langkah-langkah 1-6 pada Bab 16 sebagai berikut:

Langkah 1:
Spesifikasi filter digital:

LPF	HPF	BPF	BSF
δ_p, δ_s *(dalam dB),* f_c , f_{stop}, f_s 	δ_p, δ_s *(dalam dB),* f_c , f_{stop}, f_s 	δ_p, δ_s *(dalam dB),* f_1 , f_2 , f_3 , f_4, f_s 	δ_p, δ_s *(dalam dB),* f_1 , f_2 , f_3 , f_4, f_s 

Langkah 2:
Spesifikasi filter digital dalam radian:

LPF	HPF	BPF	BSF
δ_p, δ_s , $\omega_c = 2\pi (f_c)$ $\omega_{stop} = 2\pi (f_{stop})$	δ_p, δ_s , $\omega_c = 2\pi (f_c)$ $\omega_{stop} = 2\pi (f_{stop})$	δ_p, δ_s , $\omega_1 = 2\pi (f_2)$ $\omega_2 = 2\pi (f_2)$ $\omega_3 = 2\pi (f_3)$ $\omega_4 = 2\pi (f_4)$ $B = \omega_2 - \omega_1$ $\omega_o = \sqrt{\omega_1 \omega_2}$	δ_p, δ_s , $\omega_1 = 2\pi (f_2)$ $\omega_2 = 2\pi (f_2)$ $\omega_3 = 2\pi (f_3)$ $\omega_4 = 2\pi (f_4)$ $B = \omega_2 - \omega_1$ $\omega_o = \sqrt{\omega_1 \omega_2}$

Langkah 3:
Filter analog didesain melalui prototype filter. Semua prototype filter analog dalam bentuk LPF dengan frekuensi cut-off pada 1 rad/s. Untuk itu, spesifikasi filter analog yang diperoleh di langkah ke 2 harus ditransfer ke bentuk dasar prototype (LPF 1 rad/s). Proses transfer adalah:

LPF	HPF	BPF	BSF
δ_p, δ_s , $\Omega_c = 1$ $\Omega_{stop} = \dfrac{\omega_{stop}}{\omega_c}$	δ_p, δ_s , $\Omega_c = 1$ $\Omega_{stop} = \dfrac{\omega_c}{\omega_{stop}}$	δ_p, δ_s , $\Omega_c = 1$ $\Omega_{stop} = \dfrac{\omega_4 - \omega_3}{\omega_2 - \omega_1}$	δ_p, δ_s , $\Omega_c = 1$ $\Omega_{stop} = \dfrac{\omega_2 - \omega_1}{\omega_4 - \omega_3}$

Langkah 4:
Langkah ini adalah menentukan orde dari filter dan tranasfer function dari filter prototype. Langkah ke-4 ini persis sama dengan langkah ke-5 (untuk Butterworth, Chebyshev Tipe I, Chebyshev Tipe II, dan Elliptic) pada metode transformasi bilinier seperti pada Modul 15.

Langkah 5:
Transformasikan kembali prototype LPF 1 rad/s tersebut ke filter analog yang seharusnya (LPF atau HPF atau BPF atau BSF) dengan frekuensi sesuai dengan ω_c, ω_{stop}, ω_1, ω_2, ω_3, dan ω_4.

LPF	HPF	BPF	BSF
Gantilah s dengan $\left(\frac{1}{\omega_c}\right)s$	*Gantilah s dengan* $\left(\frac{\omega_c}{s}\right)$	*Gantilah s dengan* $\left(\frac{s^2+\omega_0^2}{Bs}\right)$	*Gantilah s dengan* $\left(\frac{Bs}{s^2+\omega_0^2}\right)$

Setelah langkah ini maka kita telah memperoleh transfer function filter analog, *H(s)*, sesuai dengan spesifikasi pada langkah ke-2

Langkah 6:
Pada langkah ini kita akan melakukan proses mencari pecahan parsial sehingga H(s) menjadi:

$$H(s)=\sum_{k=1}^{N}\frac{c_k}{s-p_k}$$

kemudian kita transformasikan menjadi:

$$H(z)=\sum_{k=1}^{N}\frac{T_s\,c_k}{1-e^{p_kT_s}z^{-1}}$$

di mana T_s adalah periode sampling.

17.2 Contoh Desain Filter IIR dengan Impulse Invariant

Sebagai contoh, kita akan mendesain suatu filter IIR Chebyshev tipe I dengan spesifikasi sebagai berikut:

Langkah 1:

Tipe filter	: LPF, Chebyshev Tipe I
Passband ripple, δ_p	: 0.5 dB
Frekuensi cut-off, f_c	: 2 kHz
Orde filter, N	: 2
Frekuensi sampling, f_s	: 10 kHz

Langkah 2: spesifikasi filter dalam radian

Frekuensi cut-off,

$\omega_c = 2\pi(2kHz) = 12566$ rad/s

Frekuensi stopband, ω_{stop} tidak perlu dihitung karena kita tidak mengetahui f_{stop}, dan juga karena kita telah mengetahui orde dari filter, *N*.

Langkah 3: frekuensi prototype

Frekuensi cut, $\Omega_c = 1$

Frekuensi stopband, Ω_{stop} juga tidak perlu dihitung dengan alasan yang sama seperti di atas.

Langkah 4: orde filter

Orde dari filter telah diketahui yaitu: $N = 2$

jadi kita menggunakan prototype filter dengan orde 2, maka

$$H(s) = \frac{1.4314}{s^2 + 1.4256s + 1.5162}$$

Langkah 5: transformasi prototype tersebut ke filter LPF dengan

$\omega_c = 12566\ rad/s$ dengan mengganti semua s dengan

$$\left(\frac{1}{\omega_c}\right)s = \left(\frac{1}{12566}\right)s = 7.958 \cdot 10^{-5} s$$

$$H(s) = \frac{1.4314}{(7.958 \cdot 10^{-5} s)^2 + 1.4256(7.958 \cdot 10^{-5} s) + 1.5162}$$

$$H(s) = \frac{1.4314}{6.3329 \cdot 10^{-9} s^2 + 1.1345 \cdot 10^{-4} s + 1.5162}$$

Langkah 6: memisahkan *H(s)* menjadi pecahan parsial:

$$H(s) = \frac{1.4314}{6.3329 \cdot 10^{-9} s^2 + 1.1345 \cdot 10^{-4} s + 1.5162} = \frac{2.2602 \cdot 10^8}{s^2 + 1.7914 \cdot 10^4 s + 2.3941 \cdot 10^8}$$

$$= \frac{2.2602 \cdot 10^8}{(s + 0.8957 \cdot 10^4 + 1.2617 \cdot 10^4 j)\ (s + 0.8957 \cdot 10^4 - 1.2617 \cdot 10^4 j)}$$

$$= \frac{A}{(s + 0.8957 \cdot 10^4 + 1.2617 \cdot 10^4 j)} + \frac{B}{(s + 0.8957 \cdot 10^4 - 1.2617 \cdot 10^4 j)}$$

Untuk mencari A dan B maka kita menyamakan penyebut sehingga

$$A\,(s+0.8957\cdot10^4-1.2617\cdot10^4\,j)+B\,(s+0.8957\cdot10^4+1.2617\cdot10^4\,j)=2.2602\cdot10^8$$

$$(A+B)s+0.8957\cdot10^4(A+B)+1.2617\cdot10^4\,j\,(B-A)=2.2602\cdot10^8$$

dari persamaan ini kita dapatkan bahwa:

$(A+B)=0$ atau $A=-B$

dan

$1.2617\cdot10^4\,j\,(B-A)=2.2602\cdot10^8$ atau

$1.2617\cdot10^4\,j\,(B-(-B))=2.2602\cdot10^8$

$2.5234\cdot10^4\,B\,j=2.2602\cdot10^8$

maka

$$B=\frac{2.2602\cdot10^8}{2.5234\cdot10^4\,j}=-0.8957\cdot10^4\,j \quad \text{dan}$$

$$A=0.8957\cdot10^4\,j$$

sehingga

$$H(s)=\frac{0.8957\cdot10^4\,j}{(s+0.8957\cdot10^4+1.2617\cdot10^4\,j)}+\frac{-0.8957\cdot10^4\,j}{(s+0.8957\cdot10^4-1.2617\cdot10^4\,j)}$$

Selanjutnya kita akan mentransfer ke *H(z)* dengan $T_s = 1/f_s = 1x10^{-4}$

$$H(z)=\frac{10^{-4}(0.8957\cdot10^4\,j)}{1-e^{(-0.8957-1.2617\,j)}z^{-1}}+\frac{10^{-4}(-0.8957\cdot10^4\,j)}{1-e^{(-0.8957+1.2617\,j)}z^{-1}}$$

untuk menyederhanakan *H(z)* maka:

$$e^{(-0.8957+1.2617j)} = e^{(-0.8957)}\ e^{(1.2617)j} = 0.4083e^{(1.2617)j} = 0.4083\angle 1.2617 = 0.1242 + 0.3890j$$

$$e^{(-0.8957-1.2617j)} = e^{(-0.8957)}\ e^{(-1.2617)j} = 0.4083e^{(-1.2617)j} = 0.4083\angle -1.2617 = 0.1242 - 0.3890j$$

maka

$$H(z) = \frac{0.8957j}{1-(0.1242-0.3890j)z^{-1}} + \frac{-0.8957j}{1-(0.1242+0.3890j)z^{-1}}$$

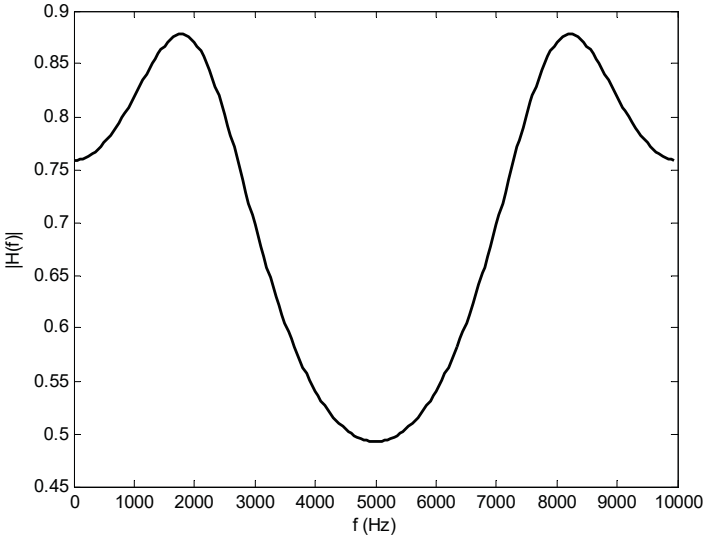
$$H(z) = \frac{0.8957j\left(1-(0.1242+0.3890j)z^{-1}\right) - 0.8957j\left(1-(0.1242-0.3890j)z^{-1}\right)}{\left(1-(0.1242-0.3890j)z^{-1}\right)\left(1-(0.1242+0.3890j)z^{-1}\right)}$$

$$H(z) = \frac{\left(0.8957j - 0.1112jz^{-1} + 0.3484z^{-1}\right) - \left(0.8957j - 0.1112jz^{-1} - 0.3484z^{-1}\right)}{\left((1-0.1242z^{-1})^2 - (0.3890jz^{-1})^2\right)}$$

$$H(z) = \frac{0.6969z^{-1}}{\left((1-0.2482z^{-1}+0.0154z^{-2}) - (-0.1513z^{-2})\right)}$$

$$H(z) = \frac{0.6969z^{-1}}{\left(1-0.2482z^{-1}+0.1667z^{-2}\right)}$$

Respon frekuensi dari filter tersebut adalah seperti pada Gambar 17.4.



Gambar 17.4
Respon frekuensi dari filter LPF hasil desain

Respon frekuensi dari filter hasil desain tersebut digambarkan dengan:

```
>> B =[0    0.6969    0];
>> A =[1.0000   -0.2482    0.1667];
>> [H,W] = freqz(B,A,200,1,'whole');
>> plot(W*10000,abs(H),'Color',[0 0 0],'LineWidth',2)
>> xlabel('f (Hz)');
>> ylabel('|H(f)|');
```

17.3 Desain Filter IIR dengan Impulse Invariant pada Matlab

Jika kita ingin mengunakan Matlab untuk melakukan proses desain dari contoh tersebut maka kita menggunakan perintah ‘impinvar’. Perintah ini mendapat input berupa H(s):

$$H(s) = \frac{2.2602 \cdot 10^8}{s^2 + 1.7914 \cdot 10^4 s + 2.3941 \cdot 10^8}$$

```
>> fs=10000;
>> B=[0 0 2.2602e8];
>> A =[1 1.7914e4 2.3894e8];
>> [bz,az]=impinvar(B,A,fs)
```

Perintah tersebut menghasilkan

```
bz =
        0    0.6974         0

az =
   1.0000   -0.2499    0.1667
```

artinya

$$H(z) = \frac{0.6974 z^{-1}}{\left(1 - 0.2499 z^{-1} + 0.1667 z^{-2}\right)}$$

Hasil ini sangat mirip dengan hasil perhitungan kita di atas. Perbedaan yang ada disebabkan oleh pembulatan angka pada saat perhitungan.

SOAL LATIHAN

1. Dengan metode impulse invariant, rancanglah suatu filter

Tipe filter	: HPF, Butterworth
Passband ripple, δ_p	: 3 dB
Stopband attenuation	: 10 dB
Frekuensi cut-off, f_c	: 2 kHz
Frekuensi stoband, f_{stop}	: 1 kHz
Frekuensi sampling, f_s	: 10 kHz

2. Amati respon frekuensi dari filter yang dirancang pada soal nomor 1 di atas. Apakah sesuai dengan spesifikasi rancangan?

3. Dengan metode impulse invariant, rancanglah suatu filter

Tipe filter	: LPF, Chebychev Type 1
Passband ripple, δ_p	: 0.1 dB
Orde	: 2
Frekuensi cut-off, f_c	: 2 kHz
Frekuensi stoband, f_{stop}	: 1 kHz
Frekuensi sampling, f_s	: 10 kHz

4. Amati respon frekuensi dari filter yang dirancang pada soal nomor 3 di atas. Apakah sesuai dengan spesifikasi rancangan?

Daftar Pustaka

John G. Proakis, Dimitris G. Manolakis, 'Digital Signal Processing: Principles, Algorithms, and, Applications', Prentice-Hall International, 2006

Li Tan, 'Digital Signal Processing: Fundamentals and Applications', Academic Press, 2008

B. Preetham Kumar, 'Digital Signal Processing Laboratory', Taylor & Francis, 2005

Lawrence R. Rabiner, Ronald W. Schafer, 'Theory and Applications of Digital Speech Processing', Prentice Hall, 2011

Nemuel D. Pah, 'Measuring Similarity between Wavelet Function and Transient in a Signal with Symmetric Distance Coefficient', The Int. Conf. on Industrial and Appl. Math., Bandung-Indonesia 2010

Mata kuliah pemrosesan sinyal digital selalu dianggap sebagai mata kuliah yang sulit baik bagi mahasiswa maupun dosen. Ada sangat banyak buku teks tentang pemrosesan sinyal digital yang membahas topik ini dengan sangat lengkap dan detail. Kelengkapan dan kedalaman dari buku-buku tersebut justru membuat kesulitan dalam mempersiapkan kuliah. Terlalu banyak bahan yang harus diajarkan, terlalu banyak rumus yang harus diturunkan dan dibuktikan, terlalu banyak konsep yang harus dipahami, terlalu banyak asumsi yang harus diterima, terlalu banyak simbol yang harus dihafal. Kondisi inilah yang membuat pemrosesan sinyal digital terlihat sangat menyeramkan.

Buku ini ditulis untuk menghadirkan suatu bahan ajar mata kuliah pemrosesan sinyal digital pada jenjang sarjana yang sederhana,padat, ringkas, dan aplikatif. Buku ini ditulis agar rekan-rekan dosen dapat dengan cepat mempersiapkan bahan ajar untuk mata kuliah ini. Buku ini juga ditujukan kepada mahasiswa yang ingin belajar dan memahami materi pemrosesan sinyal digital dengan cepat dan lengkap tanpa perlu terjebak dalam proses pembuktian rumus yang rumit. Buku ini lahir dari suatu pengalaman panjang penulis dalam mengajarkan materi ini di Jurusan Teknik Elektro Universitas Surabaya. Kiranya buku ini dapat membantu dosen dan mahasiswa memahami pemrosesan sinyal digital.

PEMROSESAN SINYAL DIGITAL

Nemuel Daniel Pah adalah dosen pada Jurusan Teknik Elektro, Universitas Surabaya. Sebagai seorang dosen dan peneliti, penulis telah membuat banyak modul kuliah dan terlibat dalam banyak penelitian di bidang pemrosesan sinyal digital, pemrosesan citra digital, dan robotika. Penulis merupakan lulusan dari Teknik Elektro ITS pada tahun 1994 dan mendapat gelar Master of Engineering (M.Eng) dan Doctor of Philosophy (Ph.D) dari RMIT University, Australia pada tahun 1999 dan 2003 dalam bidang biomedical signal processing.

ISBN: 978-602-262-789-0

GRAHA ILMU