ALGORITMA FAST FOURIER TRANSFORM (FFT) DECIMATION IN TIME (DIT) DENGAN RESOLUSI 1/10 HERTZ

Sugeng Riyanto, Agus Purwanto, Supardi
Laboratorium Riset Komputasi, Jurusan Pendidikan Fisika FMIPA UNY
Kampus Karangmalang Yogyakarta 55281

Abstrak

Fenomena kebocoran sinyal selalu muncul pada DFT dan FFT. Penggunaan FFT pada beberapa *software* seperti MATLAB 6.5[1] dan Sound Forge 6.0[2] masih memiliki keterbatasan resolusi yaitu sebesar 1 Hz. Keterbatasan resolusi diatasi dengan menggunakan fungsi *windows*. Namun demikian sinyal hasil *treatment* belum sesuai dengan kenyataan atau frekuensi sesungguhnya yang dibawa dan tergantung pada pemilihan jenis fungsi *windows.* Tujuan dari penelitian ini adalah mengetahui persamaan DFT dan FFT dengan resolusi yang lebih tinggi yaitu 1/10 Hz, kemudian mengimplementasikan persamaan tersebut ke dalam suatu program.

Metode yang digunakan dalam penelitian ini adalah komputasi numerik. Hasil penelitian menunjukkan bahwa resolusi yang lebih tinggi ditentukan oleh faktor f_s/N. Implementasi algoritma dan listing program DFT dan FFT dengan resolusi 1/10 Hz membawa konsekuensi $N=10f_s$. Secara eksperimen dengan menambah N berarti memperlama waktu menyampling dari suatu sinyal. Lamanya waktu transformasi dapat diatasi dengan meningkatkan kelas Radix FFT yang lebih tinggi.

Key words: DFT, Resolusi, Radix FFT, *Windows.*

A. PENDAHULUAN

Proses penting dalam *Digital Signal Processing* (DSP) adalah menganalisis suatu sinyal input maupun output untuk mengetahui karakteristik sistem fisis tertentu. Proses analisis dan sintesis dalam domain waktu memerlukan analisis cukup panjang dengan melibatkan turunan dari fungsi, yang dapat menimbulkan ketidaktelitian hasil analisis. Analisis dan sintesis sinyal akan lebih mudah dilakukan pada domain frekuensi, karena besaran yang paling menentukan suatu sinyal adalah frekuensi. Oleh karena itu, untuk dapat bekerja pada domain frekuensi dibutuhkan suatu formulasi yang tepat sehingga proses manipulasi sinyal sesuai dengan kenyataan.

Salah satu formulasi yang ampuh untuk proses pengolahan sinyal adalah menggunakan *Discrete Fourier Transform* (DFT). Prinsip DFT adalah mentransformasikan (alih bentuk) sinyal yang semula analog menjadi diskret dalam domain waktu, dan kemudian diubah ke dalam domain frekuensi. Hal ini dilakukan dengan mengalikan sinyal diskret dengan suatu fungsi kernel.

Algoritma lain yang lebih cepat adalah *Fast Fourier Transform* (FFT). Prinsip kerja FFT adalah membagi sinyal hasil penyamplingan menjadi beberapa bagian yang kemudian masing-masing bagian diselesaikan dengan algoritma yang sama dan hasilnya dikumpulkan kembali. Ada tiga kelas FFT yang umum digunakan di dalam suatu *software* DSP yaitu *Decimation in Time* (DIT), *Decimation in Frequency* (DIF) dan *Split Radix*. Ide ketiga jenis FFT tersebut adalah proses iterasi *sequence data* dilakukan secara berbeda dan memanfaatkan fungsi kernel yang memiliki sifat yang simetris pada suatu nilai tertentu dalam satu periode suatu sinyal. Jenis lain FFT yang sudah digunakan adalah paralel FFT dimana *sequence data* dikerjakan dengan menggunakan *parallel computin*g sehingga proses transformasi akan lebih cepat.

Hingga saat ini, penggunaan FFT pada beberapa *software* seperti MATLAB 6.5[1] dan Sound Forge 6.0[2] masih memiliki keterbatasan resolusi yaitu sebesar 1 Hz. Resolusi dapat diartikan sebagai daya pisah atau sensitifitas dari suatu alat ukur dalam memperoleh hasil ukur yang terbaik sehingga dapat membedakan perubahan terkecil dari besaran fisis. DFT dan FFT memiliki resolusi sebesar f_s/N yang mana f_s adalah *sampling rate* (dalam 1 detik diambil sebanyak f_s data) dan N adalah banyaknya data hasil penyamplingan. Umumnya keterbatasan resolusi ini diatasi dengan menggunakan fungsi *windows*. Namun demikian sinyal hasil *treatment* ini belum sesuai dengan kenyataan atau frekuensi sesungguhnya yang dibawa dan tergantung pada pemilihan jenis fungsi

windows. Oleh karena itu, sangat penting menyusun persamaan DFT dan FFT dengan resolusi yang lebih tinggi dan kemudian mengimplementasikan persamaan tersebut ke dalam suatu program.

B. KAJIAN TEORI

1. Transformasi Fourier, DFT dan FFT

a) Transformasi Fourier

Transformasi Fourier didefinisikan sebagai

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft}\, dt$$

$$= \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt - j\int_{-\infty}^{\infty} x(t)\sin(2\pi ft)dt \qquad (1)$$

dimana

$x(t)$ = fungsi atau sinyal dalam domain waktu,

$e^{-j2\pi ft}$ = fungsi kernel,

$X(f)$ = fungsi dalam domain frekuensi,

f = frekuensi.

Persamaan (1) digunakan untuk mentransformasikan sinyal dari domain waktu ke dalam domain frekuensi. Dengan keterbatasan biaya eksekusi pada komputer, maka persamaan (1), khususnya bagian real, didekati dengan

$$\int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt \rightarrow \sum_n x(n\Delta t)\cos(2\pi fn\Delta t)\Delta t$$

$$= \sum_n x(n\Delta t)\cos(2\pi nm\Delta t\Delta f)\Delta t$$

$$= \sum_n x(n\Delta t)\cos(2\pi \frac{nm}{N})\Delta t \qquad (2)$$

dengan m dan n adalah bilangan bulat.

Dalam domain waktu periode suatu sinyal dinyatakan sebagai $T = N\Delta t$, sedangkan pada domain frekuensi $\Delta f = \frac{f_s}{N}$ dengan Δf menyatakan interval antar frekuensi dan $f_s = \frac{1}{\Delta t} = N\Delta f$. Dengan demikian, dalam persamaan (2) $\Delta t\Delta f = \frac{1}{N}$, yang merupakan penghubung antara domain waktu dengan domain frekuensi. Bila jumlah data lebih kecil dari f_s maka frekuensi yang dihasilkan tidak presisi. Disisi lain f_s haruslah $\geq 2f_{maksimum}$ untuk menghindari *aliasing* frekuensi di dekat frekuensi yang dicari. *Aliasin*g merupakan fenomena munculnya frekuensi yang sama dari hasil transformasi yang mana kita tidak bisa membedakan antara frekuensi yang asli dengan frekuensi bayangan.

Pada umumnya, transformasi Fourier menggunakan alat yang disebut *real-time spectrum analyzer* yang telah terintegrasi dalam bentuk *chip* untuk menghitung sinyal diskret dalam domain waktu yang berasal dari *microphone*. Untuk dapat menganalisis spektrum frekuensi, di dalam prosessor DSP disusun program *Discrete Fourier Transform* (DFT) (Schuler, 2003: 477).

b) Discrete Fourier Transform (DFT)

Discrete Fourier Transform (DFT) didefinisikan sebagai

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)mn}, \qquad (3)$$

dengan n = *indeks dalam domain waktu* = 0, 1, ..., N-1,

m = *indeks dalam domain frekuensi* = 0, 1, ..., N-1,

Persamaan ini menyatakan bahwa DFT merupakan metode yang berguna dalam menentukan amplitudo dan komponen-komponen frekuensi harmonik ke-*m* dari suatu sinyal periodik atau merupakan koefisien-koefisien deret Fourier.

c) Fast Fourier Transform (FFT)

Berawal dari DFT-*N* data,

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi mn/N} \tag{4}$$

$x(n)$ dipilah menjadi genap dan ganjil sehingga persamaan (4) menjadi

$$X(m) = \sum_{n=0}^{(\frac{N}{2})-1} x(2n) e^{-j2\pi(2n)m/N} + \sum_{n=0}^{(\frac{N}{2})-1} x(2n+1) e^{-j2\pi(2n+1)m/N}$$

$$= \sum_{n=0}^{(\frac{N}{2})-1} x(2n) e^{-j2\pi(2n)m/N} + e^{-j2\pi m/N} \sum_{n=0}^{(\frac{N}{2})-1} x(2n+1) e^{-j2\pi(2n)m/N} \tag{5}$$

Dengan mendefinisikan $W_N = e^{-j2\pi/N}$, persamaan (5) menjadi

$$X(m) = \sum_{n=0}^{(\frac{N}{2})-1} x(2n) W_N^{2nm} + W_N^m \sum_{n=0}^{(\frac{N}{2})-1} x(2n+1) W_N^{2nm} \tag{6}$$

Karena $W_N^2 = e^{-j(2\pi/N)2} = e^{-j\frac{2\pi}{(\frac{N}{2})}}$, maka $W_N^2 = W_{\frac{N}{2}}$. Jadi persamaan (6) menjadi

$$X(m) = \sum_{n=0}^{(\frac{N}{2})-1} x(2n) W_{\frac{N}{2}}^{nm} + W_N^m \sum_{n=0}^{(\frac{N}{2})-1} x(2n+1) W_{\frac{N}{2}}^{nm} \tag{7}$$

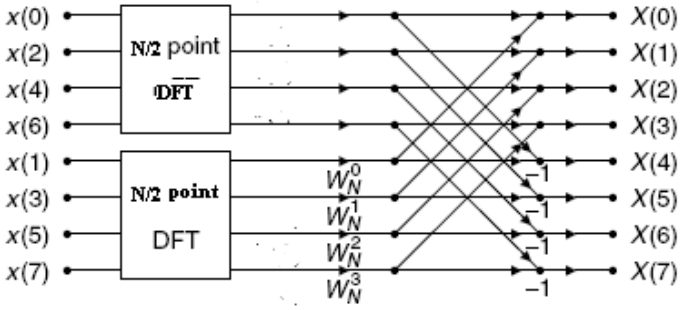
Setelah domain waktu dibagi dua, domain frekuensi juga dibagi menjadi dua yaitu

$$X(m+N/2) = \sum_{n=0}^{(\frac{N}{2})-1} x(2n) W_{\frac{N}{2}}^{n(m+N/2)} + W_N^{(m+N/2)} \sum_{n=0}^{(\frac{N}{2})-1} x(2n+1) W_{\frac{N}{2}}^{n(m+N/2)}$$

$$= \sum_{n=0}^{(\frac{N}{2})-1} x(2n) W_{\frac{N}{2}}^{nm} - W_N^m \sum_{n=0}^{(\frac{N}{2})-1} x(2n+1) W_{\frac{N}{2}}^{nm}. \tag{8}$$

Persamaan (7) dan (8) merupakan *FFT radix*-2 *Decimation in Time* (DIT) yang mana *sequence* data dipilah menjadi dua bagian menjadi genap dan ganjil dan menggambarkan gabungan dua DFT *N*/2 data. Penggunaan sifat periodik dari fungsi kernel membuat perhitungan menjadi lebih efisien karena cukup mengganti tanda operasi menjadi minus.

Secara sederhana persamaan (7) dan (8) digambarkan menggunakan diagram kupu-kupu (*butterfly diagram*) yaitu:



Gambar 1. Diagram kupu-kupu (*butterfly diagram*) FFT Radix-2 DIT (*Decimation in Time*). (Dikutip dari Li Tan, *Digital Signal Processing*, 2008: 129).

Selanjutnya akan dirumuskan FFT radix-4, dengan cara DFT-*N* data dibagi menjadi empat bagian sebagai berikut:

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

$$X(m)=\sum_{n=0}^{\frac{N}{4}-1}x(4n)W_N^{(4n)m}+\sum_{n=0}^{\frac{N}{4}-1}x(4n+1)W_N^{(4n+1)m}$$

$$+\sum_{n=0}^{\frac{N}{4}-1}x(4n+2)W_N^{(4n+2)m}+\sum_{n=0}^{\frac{N}{4}-1}x(4n+3)W_N^{(4n+3)m} \tag{9}$$

$$=\underbrace{\sum_{n=0}^{\frac{N}{4}-1}x(4n)W_N^{(4n)m}}_{Y(m)}+W_N^m\underbrace{\sum_{n=0}^{\frac{N}{4}-1}x(4n+1)W_N^{(4n)m}}_{Z(m)}$$

$$+W_N^{2m}\underbrace{\sum_{n=0}^{\frac{N}{4}-1}x(4n+2)W_N^{(4n)m}}_{G(m)}+W_N^{3m}\underbrace{\sum_{n=0}^{\frac{N}{4}-1}x(4n+3)W_N^{(4n)m}}_{H(m)} \tag{10}$$

Jika kemudian indeks domain frekuensi dibagi menjadi empat bagian dengan *m* = 0, 1, 2, ...*N*/4-1 maka persamaan (10) menjadi

a) $X(m)=Y(m)+W_N^mZ(m)+W_N^{2m}G(m)+W_N^{3m}H(m)$ (11)

b) $X(m+N/4)=Y(m)+W_N^{m+N/4}Z(m)$

$$+W_N^{2(m+N/4)}G(m)+W_N^{3(m+N/4)}H(m) \tag{12}$$

c) $X(m+N/2)=Y(m)+W_N^{m+N/2}Z(m)$

$$+W_N^{2(m+N/2)}G(m)+W_N^{3(m+N/2)}H(m) \tag{13}$$

d) $X(m+3N/4)=Y(m)+W_N^{m+3N/4}Z(m)$

$$+W_N^{2(m+3N/4)}G(m)+W_N^{3(m+3N/4)}H(m). \tag{14}$$

Dengan menggunakan faktor *twiddle* yaitu:

$W_N^{\frac{N}{2}}=e^{-j2\pi\frac{(N/2)}{N}}=e^{-j\pi}=-1, W_N^{\frac{N}{4}}=W_4=e^{-j\frac{2\pi}{4}}=-j$ dan $W_N^{\frac{3N}{4}}=(-j)^3=j$

maka FFT Radix-4 DIT secara keseluruhan dapat dituliskan sebagai

$$X(m)=[Y(m)+W_N^{2m}G(m)]+[W_N^mZ(m)+W_N^{3m}H(m)] \tag{15}$$

$$X(m+N/4)=[Y(m)-W_N^{2m}G(m)]-j[W_N^mZ(m)-W_N^{3m}H(m)] \tag{16}$$

$$X(m+N/2)=[Y(m)+W_N^{2m}G(m)]-[W_N^mZ(m)+W_N^{3m}H(m)] \tag{17}$$

$$X(m+3N/4)=[Y(m)-W_N^{2m}G(m)]+j[W_N^mZ(m)-W_N^{3m}H(m)] \tag{18}$$

Dengan cara yang sama, kelas Radix-DIT yang lebih tinggi dapat dirumuskan dengan cara sebagai berikut:

Diawali dengan DFT *N*-data yaitu:

$$X(r) = \sum_{l=0}^{N-1} x(l) W_N^{rl} \tag{19}$$

$$= \sum_{u=0}^{q-1} \sum_{k=0}^{\frac{N}{q}-1} x(qk+u) W_N^{r(qk+u)}$$

$$= \sum_{u=0}^{q-1} W_N^{ur} \sum_{k=0}^{\frac{N}{q}-1} x(qk+u) W_N^{r(qk)}$$

$$= \sum_{u=0}^{q-1} W_N^{ur} \sum_{k=0}^{\frac{N}{q}-1} x(qk+u)(W_N^q)^{kr} = \sum_{u=0}^{q-1} W_N^{ur} \sum_{k=0}^{\frac{N}{q}-1} x(qk+u) W_{\frac{N}{q}}^{kr} \tag{20}$$

Dengan mensubstitusikan $r = l + \lambda \frac{N}{q}$ maka: (21)

$$X(l+\lambda\frac{N}{q}) = \sum_{u=0}^{q-1} W_N^{u(l+\lambda\frac{N}{q})} \sum_{k=0}^{\frac{N}{q}-1} x(qk+u) W_{\frac{N}{q}}^{k(l+\lambda\frac{N}{q})}$$

$$= \sum_{u=0}^{q-1} W_N^{ul} (W_N^{\frac{N}{q}})^{u\lambda} \sum_{k=0}^{\frac{N}{q}-1} x(qk+u) W_{\frac{N}{q}}^{k(l+\lambda\frac{N}{q})}$$

$$X(l+\lambda\frac{N}{q}) = \sum_{u=0}^{q-1} W_N^{ul} W_q^{u\lambda} \left(\sum_{k=0}^{\frac{N}{q}-1} x(qk+u) W_{\frac{N}{q}}^{kl} \right) \tag{22}$$

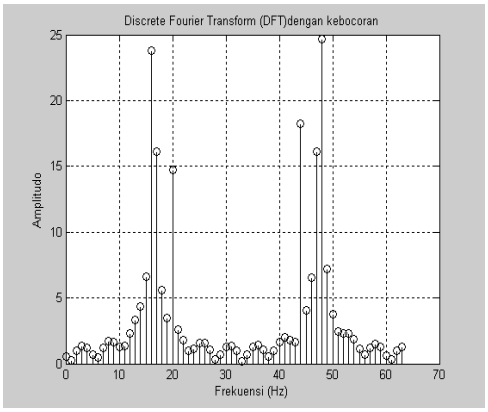
dengan
q = jenis Radix =2^m , m = 1, 2, 3,…
λ = 0, 1, 2, …q-1, atau 0: 1: q-1,
N= banyaknya data,
l = indeks dalam domain frekuensi= 0, 1, 2, … $\frac{N}{q/2} - 1$,
u =0, 1, 2, …q-1,
k = indeks dalam domain waktu= 0, 1, 2,… $\frac{N}{q} - 1$

(Chu and George, 2000: 21-25).

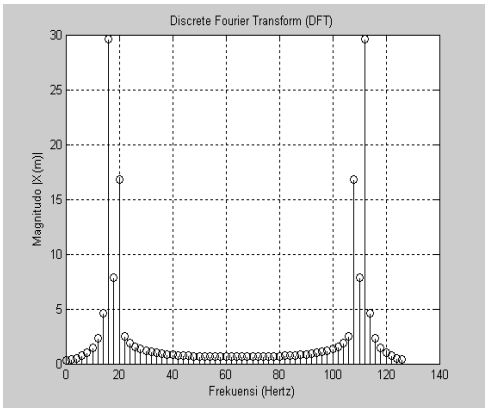
2. Windowing

Fenomena kebocoran sinyal muncul ketika frekuensi dari sinyal input bukan merupakan bilangan bulat dan $N \neq f_s$, seperti tampak pada Gambar 2 berikut ini:

Fenomena kebocoran



a) Sinyal ketika frekuensi bukan bilangan bulat dengan *N* (banyaknya data penyamplingan) = 64, f_s (*sampling rate*) = 64 Hz, f_1=16,4 Hz, f_2=20 Hz. (frekuensi masukan).



Sinyal ketika *N* (banyaknya data penyamplingan) tidak sama dengan f_s (*sampling rate*). Dalam hal ini *N* = 64, f_s=128 Hz, f_1=16,4 Hz, f_2=20 Hz. (frekuensi masukan).

Gambar 2. DFT resolusi 1 Hz.

Pada prinsipnya penggunaan fungsi *windowing* adalah dengan cara melewatkan sinyal yang mempunyai frekuensi sembarang dikonvolusikan dengan fungsi window tertentu sehingga dapat mereduksi sinyal-sinyal yang tergolong bocor sebelum dilakukan proses transformasi. Ada beberapa fungsi *windows* yang telah ada diantaranya *Hann, Hamming, triangular, rectangular*.

Secara matematis penggunaan windowing pada DFT dapat dinyatakan sebagai

$$X(m) = \sum_{m=0}^{N-1} w(n) * x(n) W_N^{nm} \; . \tag{23}$$

Berikut ini beberapa jenis fungsi windows yang telah didefinisikan:

1) Hann window

$$w(n) = 0{,}5 - 0{,}5\cos(2\pi n / N - 1) \text{ untuk } n = 0,1,2,\ldots,N\text{-}1 \tag{24}$$

2) Hamming window

$$w(n) = 0{,}54 - 0{,}46\cos(2\pi n / N - 1) \text{ untuk } n = 0, 1, 2, \ldots,N\text{-}1 \tag{25}$$

3) Triangular window

$$w(n) = \frac{n}{N/2} \text{ untuk } n = 0,1,2,\ldots,N/2 \tag{26}$$

$$w(n) = 2 - \frac{n}{N/2} \text{ untuk } n = N/2+1, N/2+2, \ldots N\text{-}1. \tag{27}$$

4) Rectangular window

$$w(n) = 1 \text{ untuk } n = 0, 1, 2, \ldots,N\text{-}1. \tag{28}$$

C. METODE PENELITIAN

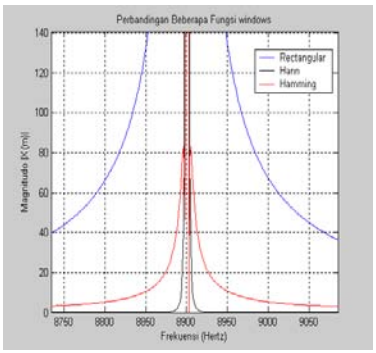
Untuk dapat mewujudkan algoritma DFT dan FFT dengan resolusi (Δf) yang lebih teliti maka ditentukan oleh faktor f_s/N, kemudian untuk menghindari efek *aliasing* pada daerah spektrum utama maka $f_s \geq 2f_{Nyquist.}$. Frekuensi audio yang didengar oleh manusia berada diantara rentang nilai frekuensi 20 Hz hingga 20.000 Hz. Oleh karena itu, f_s yang digunakan dalam penelitian ini adalah dua kali atau lebih dari f_{max} yaitu ≥ 44.100 Hz. Dalam penelitian ini metode yang digunakan adalah komputasi numerik. Hal ini mengingat rumusan matematis yang ada membutuhkan perhitungan

dan iterasi yang banyak sehingga tidak mungkin jika dilakukan perhitungan secara manual. Dalam penelitian ini alat yang digunakan adalah komputer yang telah terinstal *software* MATLAB 6.5, MAPLE 10, Ubuntu dan Microsoft Office dengan spesifikasi sebagai berikut
Operating System : Microsoft Windows XP Professional (5.1, build 2600),
Processor : AMD Athlon (tm) 64 X2 dual core Processor 4400+, MMX, 3DNow (2 CPUs),
Memory : 512 MB RAM

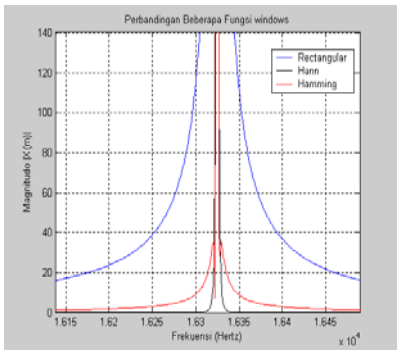
D. HASIL DAN PEMBAHASAN

1. Perbandingan frekuensi yang diperoleh dari ketiga fungsi Windows untuk Rentang Frekuensi Audio.

Perbandingan beberapa window dengan N (banyaknya data penyamplingan) = 44.100, f_s = 44.100 Hz, f_1 = 8.900,4 Hz, f_2 = 16.432,7 Hz (frekuensi masukan).



Perbandingan beberapa *window* yang diperbesar untuk f_1 (frekuensi masukan).



Perbandingan beberapa window yang diperbesar untuk f_2 (frekuensi masukan).

Gambar 3. Perbandingan penggunaan beberapa fungsi window pada rentang frekuensi audio.

Gambar 3 merupakan perbandingan beberapa fungsi *window* untuk mengatasi kebocoran sinyal dan menunjukkan bahwa pendekatan window Hann paling baik diantara fungsi window lainnya, kemudian secara berurutan adalah window Hamming dan window rectangular.
Penggunaan beberapa fungsi *windows* yang telah ada bukan merupakan solusi yang tuntas karena hanya mengurangi amplitudo dari frekuensi-frekuensi yang dianggap bocor dan mempertinggi amplitudo dan frekuensi sebenarnya. Prinsip daripada *window* adalah mengalikan DFT dengan fungsi genap sehingga tidak saling menghilangkan dan meredam amplitudo-amplitudo di sekitar frekuensi asli dari sinyal.

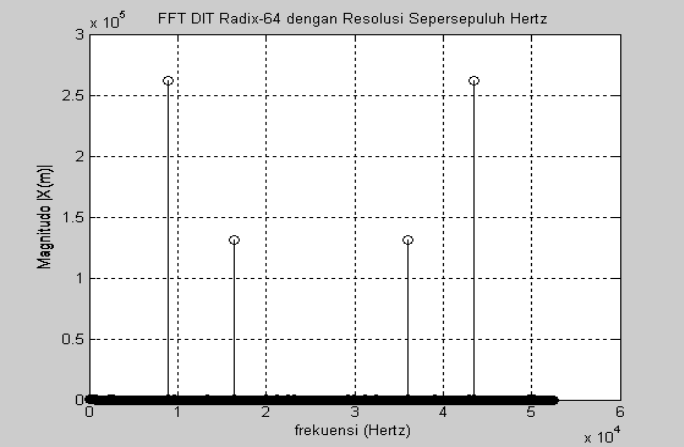
2. Perbandingan Waktu yang dibutuhkan untuk mengeksekusi Instruksi program

Tabel 1. Perbandingan waktu eksekusi antar algoritma.

No.	Algoritma	Waktu (detik)			
		Kelas	**N = f_s = 64**	**N = f_s = 512**	**N = f_s = 2048**
			f_1=10; f_2 = 20 (Hz)	**f_1 =100;f_2 = 200 (Hz)**	**f_1 = 100; f_2 = 700 (Hz)**
1	DFT	-	0,9070	43,8750	530,4060
2	FFT DIT *Radix*	2	0,1250	3,0630	24,0320
		4	**0,0470**	**0,7340**	**13,5470**
		8	0,0930	1,0780	15,9690
		16	0,2180	1,2500	15,4850
		32	0,8600	1,7820	16,0930
		64	3,4530	6,0160	20,5310
3	FFT fungsi bawaanMATLAB	-	0,0320	0,1400	3,4690

Tabel 1 menunjukkan perbandingan waktu yang dibutuhkan untuk menyelesaikan proses transformasi dari domain waktu ke dalam domain frekuensi. Pada algoritma FFT terlihat bahwa kelas FFT Radix 4 membutuhkan waktu eksekusi lebih cepat dibandingkan kelas lainnya. Hal demikian karena FFT Radix 4 membutuhkan *cost arithmetics* yang lebih sedikit dibandingkan dengan kelas Radix FFT yang lebih tinggi. FFT bawaan dari MATLAB mempunyai waktu eksekusi paling cepat karena kemungkinan besar menggunakan representasi matriks, sehingga lebih *match* dengan *software* MATLAB sendiri yang bekerja menggunakan matriks. Namun demikian ketika N dan f_s nilainya cukup besar maka fungsi bawaan dari MATLAB ini tidak mampu menjalankan perintah dengan memunculkan peringatan *out of memory*.

3. Perbandingan lamanya waktu yang dibutuhkan untuk menjalankan Instruksi DFT dan FFT dengan Resolusi 1/10 Hz



Gambar 4. FFT Radix-64 DIT dengan resolusi 1/10 Hz.

Gambar 4 adalah hasil eksekusi FFT dengan N = 524.288 dan f_s = 52.428,8 Hz. Gambar tersebut menunjukkan tidak adanya kebocoran sinyal dan tepat pada frekuensi f_1 = 8.900,3 Hz dan f_2 = 16.364,4 Hz. Hal ini terjadi karena masing-masing frekuensi memiliki tempat tertentu sehingga energi yang dibawa oleh sinyal dapat secara tepat menempati wadahnya.

Tabel 2. Perkiraan dan Perbandingan waktu eksekusi antar algoritma untuk resolusi 1/10 Hz.

No.	**Algoritma**	**Frekuensi** $\mathbf{f_1 = 8.900,3\ Hz;}$ $\mathbf{f_2 = 16.364,4\ Hz}$	**Software yang digunakan**	**Waktu**
1	DFT	N = 441.000; f_s = 44.100 Hz	MATLAB	± 8 bulan
2	FFT DIT	N = 524.288; f_s = 52.428,8 Hz	MATLAB untuk *Radix* 4	± 7 hari
			MATLAB untuk *Radix* 64	11,57 hari
			Open source: Ubuntu untuk *Radix* 4	± 10 jam
			Open source: Ubuntu untuk *Radix* 8	± 18 jam

Tabel (2) menunjukkan perbandingan waktu antara DFT dan FFT untuk menyelesaikan proses transformasi. Algoritma DFT membutuhkan waktu lebih lama karena untuk menyelesaikan satu kali transformasi DFT harus melakukan N kali perkalian dan (N-1) penjumlahan. Penggunaan

software secara tepat juga menentukan waktu lamanya eksekusi. Hal ini dapat terlihat dari hasil eksekusi antar algoritma FFT, penggunaan bahasa pemrograman C atau bahasa yang lebih rendah akan lebih cepat bila dibandingkan dengan MATLAB.

Fenomena kebocoran sinyal selalu muncul pada DFT dan FFT disebabkan oleh tidak adanya tempat bagi frekuensi yang lebih halus. Pendekatan secara diskret pada DFT dan FFT telah diperoleh sebagaimana pada Gambar 4. DFT dan FFT dengan resolusi 1/10 Hz dapat dihasilkan dengan cara memperbanyak N (data yang di*sampling*) yaitu sepuluh kali f_s. Secara eksperimen penambahan N berarti memperlama pengambilan *sampling* dari suatu sinyal. Hal ini berarti mengulangi informasi yang sama dari suatu sinyal, karena sebenarnya informasi yang sama cukup diperoleh dengan menyampling dalam waktu satu periode. Dengan menambah N, maka waktu yang dibutuhkan untuk menyelesaikan seluruh instruksi akan semakin lama. Untuk mengatasi hal demikian, maka diupayakan dengan meningkatkan *Radix* FFT sehingga proses transformasi akan lebih cepat daripada DFT. Dengan keterbatasan spesifikasi komputer yang digunakan, dalam penelitian ini baru bisa dicapai Radix 64. Hal ini terjadi karena dibatasi oleh kinerja dari komputer yang memiliki ukuran bit tertentu sehingga ketika melakukan perhitungan aritmatika bilangan *floating point* diluar jangkauan dari rentang nilai yang diizinkan, yang berakibat munculnya peringatan *out of memory* pada MATLAB.

E. KESIMPULAN

Berdasarkan pada hasil dan kajian bab-bab sebelumnya dapat disimpulkan beberapa hal sebagai berikut:

1. Perumusan DFT dan FFT dengan resolusi yang lebih teliti memenuhi persamaan f_s/N. Hal ini merupakan konsekuensi dari perumusan secara diskret pada deret Fourier.
2. Implementasi algoritma dan listing program DFT dan FFT dengan resolusi 1/10 Hz membawa konsekuensi $N=10f_s$. Secara eksperimen dengan menambah N berarti memperlama waktu menyampling dari suatu sinyal. Lamanya waktu transformasi dapat diatasi dengan meningkatkan kelas Radix FFT yang lebih tinggi.
3. Perbandingan waktu yang dibutuhkan untuk menjalankan perintah program DFT dan FFT dengan resolusi 1/10 Hz dapat dilihat pada Tabel 2. Secara numerik algoritma DFT lebih teliti karena setiap perhitungan dilakukan secara menyeluruh. Namun hal ini membawa konsekuensi waktu yang dibutuhkan lebih lama. Algoritma FFT merupakan solusi untuk mengatasi masalah waktu dengan memanfaatkan sifat unik dari fungsi kernel yang periodik pada nilai-nilai tertentu.

F. DAFTAR PUSTAKA

Chu, Eleanor, Alan George. (2000). *Inside the Fast Fourier Transform Black Box: Serial and parallel FFT Algorithms*. Boca Raton, FL: CRC Press. Hlm. 21-25.

Schuler, A. Charles. (2003). *Electronics: Principles and Applications 6thed.*. Singapore: Mc Graw Hill. Hlm. 477.

Tan, Li. (2008). *Digital Signal Processing: Fundamentals and Applications*. Singapore: Elsevier dan Academic Press. Hlm. 129.