SIMULASI PENGOLAHAN SINYAL DIJITAL FILTER ADAPTIF MENGGUNAKAN ALGORITMA LMS, RLS, FAST KALMAN, DAN GAL

Susmini Indriani Lestariningati[1)]
[1)]Jurusan Teknik Komputer Universitas Komputer Indonesia (UNIKOM) Bandung
Jl. Dipati Ukur No. 112-116, Bandung 40132
email: lestariningati@yahoo.com

ABSTRAK

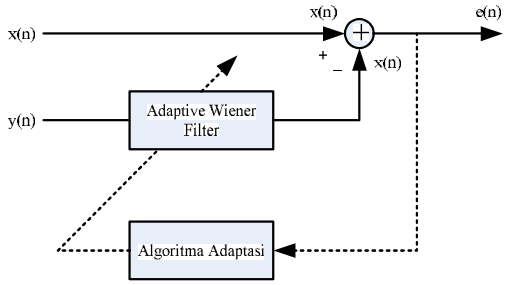
Pemrosesan sinyal dijital menggunakan filter adaptif digunakan pada banyak aplikasi terutama dalam bidang telekomunikasi, misalnya untuk menghilangkan noise dari suatu sinyal informasi. Terdapat banyak algoritma filter adaptif yang dapat digunakan, seperti algoritma LMS (Least Mean Square), RLS (Recursive Least Square) dan GAL (Gradient Adaptive Lattice). Filter adaptif dalam aplikasi real-time biasanya diimplementasikan menggunakan hardware yang dirancang untuk aplikasi pengolahan sinyal dijital (misalnya card TMS). Meskipun demikian, dapat juga dilakukan simulasi filter dijital menggunakan software pada PC.

Pada paper ini dibuat simulasi untuk jenis-jenis filter adaptif beserta karakteristiknya. Simulasi tersebut dibuat dengan alat bantu komputer dengan bahasa pemrograman Pascal untuk sistem operasi Windows dari algoritma filter adaptif yang bermacam-macam itu sendiri, dipilih untuk disimulasikan hanya algoritma yang umum dan mendasari algoritma yang lainnya ,yaitu algoritma LMS, RLS, Fast Kalman, dan GAL. Pada masing-masing algoritma yang akan dianalisis adalah kinerjanya, faktor-faktor yang mempengaruhi kinerja tersebut, dan diperbandingkan masing-masing kinerja.

Kata kunci: White Gaussian Noise, Mean Square Error, Autokorelasi.

1. PENDAHULUAN

Semua Filter Adaptif memakai filter Wiener sebagai realisasi filter optimum yaitu dengan kriteria *mean square error* minimum. Semua Algoritma dengan sejalannya waktu berusaha untuk konvergen mendekati kondisi ini. Seperti pada prediksi liniear, filter adaptif juga didasari oleh filter Wiener. Semua kaidah dan sifat-sifat yang berlaku pada filter Wiener tetap berlaku untuk aplikasi adaptif ini. Perbedaan penerapan filter Wiener untuk aplikasi non adaptif dan adaptif adalah jika pada filter Wiener data diproses dengan H optimum perblok cuplikan maka pada filter adaptif data diproses dengan H optimum percuplikan. Berarti H harus menuju optimum dan data harus diproses sebelum cuplikan selanjutnya datang, ini merupakan persoalan kecepatan konvergensi dari algoritma adaptif. Masalah yang lain adalah karena tidak ada blok cuplikan, berarti tidak bisa diterapkannya metoda *sample autocorrelation* untuk mendapatkan informasi tentang fungsi autokorelasi sebelumnya. Karena operasi dilakukan pada setiap cuplikan, maka H yang selalu stationer dan *time invariant* dalam satu blok, sekarang menjadi *time variant*.



Gambar 1. Skema Blok Filter Adaptif Wiener

Ada banyak algoritma adaptasi untuk mendapatkan H optimum tersebut.Algoritma yang paling mendasar adalah algoritma LMS (*Least Mean Square*), algoritma RLS (*Recursive Least Square*), dan Algoritma GLMS (*Gradient Least Mean Square*).Struktur filter H dapat berupa FIR (*Finite Impulse Response*) atau Lattice karena banyak algoritma adaptasi untuk struktur tersebut, sementara belum ditemukan algoritma adaptasi yang umum untuk struktur IIR (*Infinite Impulse Transform*). Orde filter dibatasi oleh *mean square error* yang diinginkan, dan kecepatan *processing* yang harus dicapai. Dengan makin besarnya orde filter tentu *mean square error*

semakin kecil tetapi kecepatan *processing* makin lambat. Jadi *trade-off* harus dilakukan dalam penentuan orde filter ini.

2. TUJUAN

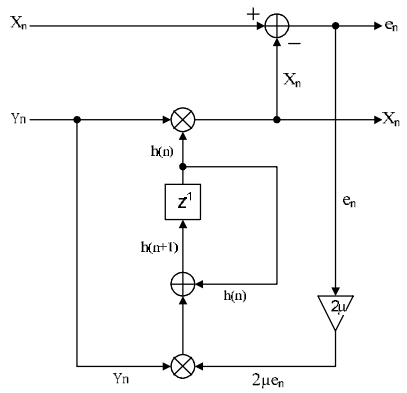
Dalam percobaan ini diamati perilaku dan kinerja 4 algoritma filter adaptif, yaitu LMS (*Least Mean Square*), RLS (*Recursive Least Square*), Fast Kalman dan GAL (*Gradient Adaptive Lattice*). Kinerja dan perilaku filter diamati dengan cara memasukkan sinyal-sinyal sinusoidal, Gaussian White Noise serta sinyal siklo-stasioner. Sinyal-sinyal ini dapat dimasukkan secara terpisah maupun gabungan.

Perilaku sinyal yang diamati meliputi *excess error*, stabilitas dan kecepatan konvergensi.

3. DASAR TEORI

3.1 Algoritma LMS (Least Mean Square)

Algorima ini berdasarkan metoda *gradient/ steepest descent* dan Widrow-Hoft LMS menghasilkan persamaan adaptasi H:



Gambar 2. Struktur Filter Adaptif LMS FIR Orde 1

Algoritma LMS dalah sebagai berikut:

1. Pada waktu n, bobot filter h(n) diketahui
2. Hitung $\hat{x}_n = \mathbf{h}(n)^T \mathbf{y}_n$
3. Hitung $e_n = x_n - \hat{x}_n$
4. Hitung bobot filter berikutnya:
   $\mathbf{h}(n+1)=\mathbf{h}(n)+2\mu e_n \mathbf{y}_n$
5. Lanjutkan ke n = n + 1

Adapun struktur filter FIR untuk orde 1 dapat dilihat pada gambar 2. Terlihat dari algoritma secara numerik akan selalu stabil untuk nilai μ tertentu.

Inisialisasi algoritma hanya diperlukan untuk menginisialisasi koefisien filter **h**. Penentuan nilai μ menentukan adanya konvergensi, kecepatan konvergensinya, dan *excess error*.

Nilai μ, terbatas diatas dan dibawah dimana menurut teori, batasnya menentukan kestabilan/ konvergensinya.

Batas nilai μ adalah $0<\mu<\frac{1}{(M+1)R}$, dimana R adalah fungsi autokorelasi signal y_n, dan M adalah orde filter FIR. Besarnya nilai μ mempengaruhi *excess error,* dan kecepatan menuju konvergensi.

Excess error sendiri dapat dicari dengan rumus:

$$\textit{Excess error} = \mu E_{min}. \qquad (1)$$

Sementara E_{min} adalah *mean square error minimum* yang dicapai pada saat $h = h_{opt}$, jadi:

$$E_{min} = E[x_n^2] - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r}$$

dimana **R** adalah matriks autokorelasi signal y_n dan **r** adalah matriks korelasi antara sinyal y_n dan sinyal x_n.

3.2 LMS signum

LMS signum dipakai untuk ADPCM (*Adaptive Differential Pulse-Code Modulation*). LMS signum adalah variasi dari LMS dengan hanya memperhitungkan polaritas e_n dan y_n, tanpa memperhitungkan nilainya. Sehingga persamaan pada algoritma LMS biasa menjadi:

$$\mathbf{h(n+1) = h(n) + 2\mu[e_n][y_n]} \qquad (2)$$

dimana $[y_n]$ atau $[e_n]$ =1 jika $e_n > 0$, $[e_n] = 0$ jika $e_n = 0$, $[e_n] = -1$ jika $e_n < 0$

Karena lebih sederhana proses perhitungannya, realisasi ke perangkat keras lebih mudah dan murah, tetapi *excess error* lebih besar dibanding LMS biasa untuk nilai μ yang sama. Masalah besarnya *excess error* tersebut dapat diatasi dengan memperkecil nilai μ.

3.3 Algoritma RLS (*Recursive Least Square*)

Adalah penerapan dari kriteria *Least Square Error* minimum.

$$LSE = \sum_n [e_n]^2 \qquad (3)$$

dan persamaan rekursif yang mempunyai tujuan untuk menghindari matriks autokorelasi yang semakin besar dengan berjalannya waktu, serta matriks inversion lemma. Persamaan dan variabel yang digunakan dengan algoritma ini lebih banyak dari pada algoritma LMS, karena lebih banyak faktor yang dipertimbangkan oleh

algoritma ini dibandingkan algoritma LMS. Algoritma RLS adalah sebagai berikut:

1. $\mathbf{k}(n/n-1) = \lambda^{-1}\boldsymbol{P}(n-1)\mathbf{y}(n)$
2. $\nu(n) = \mathbf{k}(n/n-1)^T\mathbf{y}(n)$, $\mu(n) = 1/(1+\nu(n))$
3. $\mathbf{k}(n) = \mu(n)\mathbf{k}(n/n-1)$
4. $\boldsymbol{P}(n) = \lambda^{-1}\boldsymbol{P}(n-1) - \mathbf{k}(n)\mathbf{k}(n/n-1)^T$
5. $\hat{x}(n/n-1) = \mathbf{h}(n-1)^T\mathbf{y}(n)$, $e(n/n-1)=x(n)-\hat{x}(n/n-1)$, $e(n) = \mu(n)e(n/n-1)$, $\hat{x}(n) = x(n) - e(n)$
6. $\mathbf{h}(n) = \mathbf{h}(n-1) + e(n/n-1)\mathbf{k}(n)$
7. Lanjutkan ke n = n + 1

Inisialisasi pada algoritma RLS lebih banyak dibanding algoritma LMS, untuk matriks $\boldsymbol{P}$ diinisialisasi = ∞ secara teoritis, tetapi orde 1000 sudah mencukupi secara praktis. Untuk varibel lainnya seperti k, cukup diinisialisasi = 0.

3.4 Algoritma Fast Kalman

Algoritma Fast Kalman adalah salah satu pengembangan dari algoritma RLS, dengan tujuan untuk mempercepat proses perhitungan matriks yang banyak oleh algoritma RLS. Algoritma ini tetap mempergunakan kriteria LSE minimum seperti pada algoritma RLS ditambah sebagian langkah algoritma LMS. Oleh karena itu masalah pengaruh faktor λ yang berlaku untuk algoritma RLS, berlaku juga untuk algoritma Fast Kalman.

Inisialisasi pada algoritma ini lebih banyak lagi, yaitu

$\mathbf{h}(-1) = 0$, $\mathbf{a}(-1) = [1,\mathbf{0}^T]^T$, $\mathbf{b}(-1) = [\mathbf{0}^T,1]^T$, $\tilde{k}(0/-1)=0$, dan $E^+ = \delta$, dimana δ adalah bilangan positip.

Adapun algoritma Fast Kalman adalah sebagai berikut:

0. Setiap waktu n kita mempunyai nilai-nilai h(n-1), a(n-1), b(n-1), k(n), E+(n-1), x(n), dan y(n).
1. $e^+(n/n-1) = \mathbf{a}(n-1)^T\mathbf{y}(n)$
2. $\mathbf{a}(n) = \mathbf{a}(n-1) - e^+(n/n-1)\begin{bmatrix}0\\ \tilde{\mathbf{k}}(n)\end{bmatrix}$
3. $e^+(n) = \mathbf{a}(n)^T\mathbf{y}(n)$
4. $E^+(n) = E^+(n-1) + e^+(n)\,e^+(n/n-1)$
5. Hitung elemen pertama dari $\mathbf{k}(n)$,
   $k_0(n) = \dfrac{e^+(n)}{E^+(n)}$
6. $\mathbf{k}(n) = \begin{bmatrix}0\\ \tilde{\mathbf{k}}(n)\end{bmatrix} + k_0(n)\mathbf{a}(n)$, kemudian pisahkan elemen terakhir $k_M(n)$
7. $e^-(n/n-1) = \mathbf{b}(n-1)^T\mathbf{y}(n)$
8. $\mathbf{b}(n) = \dfrac{\mathbf{b}(n-1) - e^-(n/n-1)\mathbf{k}(n)}{1-e^-(n/n-1)k_M(n)}$
9. $\begin{bmatrix}\bar{\mathbf{k}}(n)\\ 0\end{bmatrix} = \mathbf{k}(n) - k_M(n)\mathbf{b}(n)$
10. $\hat{x}(n/n-1) = \mathbf{h}(n-1)^T\mathbf{y}(n)$, $e(n/n-1) = x(n) - \hat{x}(n/n-1)$
11. $\mathbf{h}(n) = \mathbf{h}(n-1) + e(n/n-1)\mathbf{k}(n)$
12. $\hat{x}(n) = \mathbf{h}(n)^T\mathbf{y}(n)$, $e(n) = x(n) - \hat{x}(n)$
13. $\tilde{\mathbf{k}}(n+1) = \bar{\mathbf{k}}(n)$, lanjutkan untuk n berikutnya

Pada gambar 4 diperlihatkan salah satu cara penggambaran struktur filter Fast Kalman untuk orde filter = 1

3.5 Gradient Adaptive Lattice filter

Dengan menggunakan persamaan *forward* dan *backward predictor*, metoda Burg dan gradient descent, dapat dibentuk suatu algoritma yang mempercepat konvergensi algoritma LMS maupun RLS.

Algoritma Lattice RLS (LRLS) biasanya memiliki kecepatan konvergensi lebih tinggi dibandingkan Gradient Adaptive Lattice. Dalam percobaan ini akan disimulasikan algoritma *Gradient Adaptive Lattice.* Algoritma lattice pada umumnya merupakan algoritma yang paling unggul dibandingkan algoritma lainnya karena algoritma ini cepat dan stabil.

Hal yang menarik dari struktur lattice adalah adanya dua loop iterasi yaitu selain loop iterasi sample juga ada loop iterasi orde filter latticenya.

Algoritma GAL sendiri menurut pemakaian *feedback*-nya pada struktur perhitungan, dapat dibagi menjadi dua jenis yaitu GAL A dan GAL B. Tetapi belakangan ini GAL B yang lebih banyak dipakai karena *feedback* diperhitungkan setiap penambahan orde lattice sehingga kinerjanya lebih baik dibanding GAL A.

Algoritma *Gradient Adaptive Lattice* model B (GAL B) adalah sebagai berikut :

1. Pada setiap n nilai dari $\gamma_p(n)$, $d_p(n)$, untuk p=1,2,...,M dan $g_p(n)$, $d_p^-(n)$ untuk p=0,1,...,M telah diperoleh dari n sebelumnya atau dari inisialisasi.
2. Inisialisasi untuk struktur latice p=0,
   $e_0^+(n) = y_n$, $e_0^-(n) = y_n$,
   $\hat{x}_0(n) = g_0(n)e_0(n)$, $e_0(n) = x_n - \hat{x}_0(n)$

$d_0^-(n) = \lambda\, d_0^-(n-1) + e_0^-(n)^2$

$g_0(n+1) = g_0(n) + \dfrac{\beta}{d_0^-(n)}\, e_0(n) e_0^-(n)$

3. Untuk p=1,2,...,M dihitung

$e_p^+(n) = e_{p-1}^+(n) - \gamma_p(n) e_{p-1}^-(n-1)$

$e_p^-(n) = e_{p-1}^-(n-1) - \gamma_p(n) e_{p-1}^+(n)$

$d_p(n) = \lambda\, d_p(n-1) + e_{p-1}^+(n)^2 + e_{p-1}^-(n)^2$

$\gamma_p(n+1) = \gamma_p(n) + \dfrac{\beta}{d_p(n)} [\, e_p^+(n) e_{p-1}^-(n-1) + e_p^-(n) e_{p-1}^+(n)$

$\hat{x}_p(n) = \hat{x}_{p-1}(n) + g_p(n) e_p^-(n)$

$e_p(n) = e_{p-1}(n) + g_p(n) e_p^-(n)$

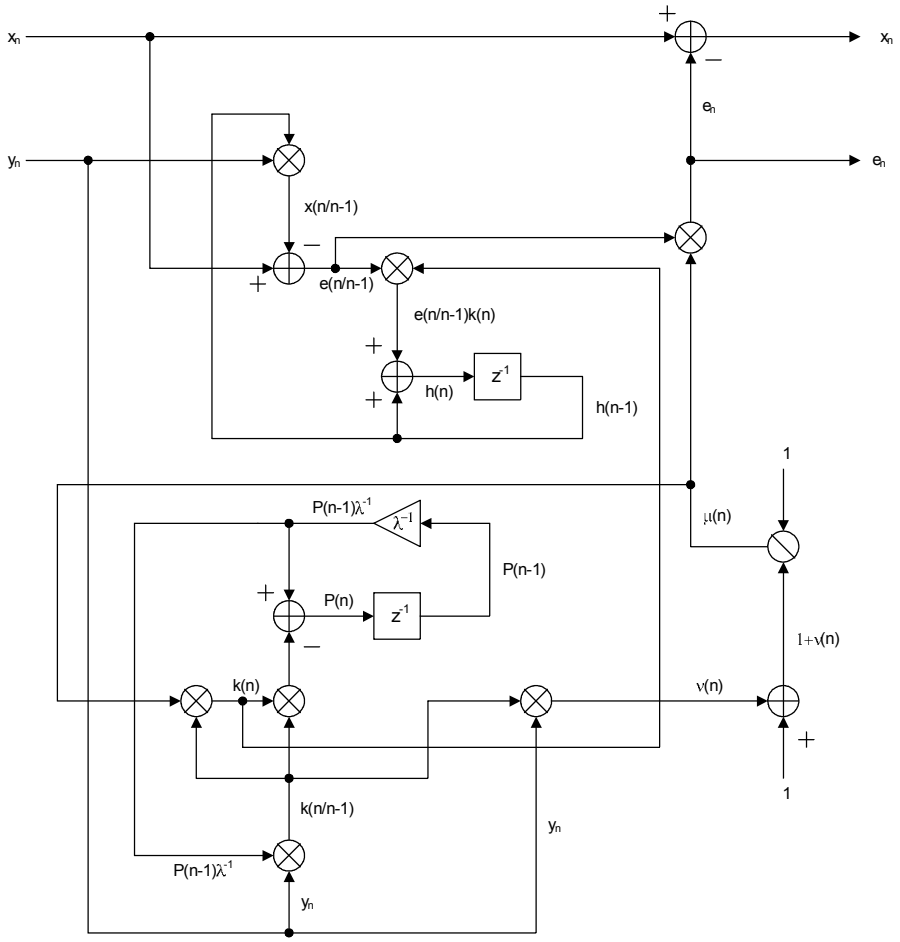
$d_p^-(n) = \lambda\, d_p^-(n-1) + e_p^-(n)^2$

$g_p(n+1) = g_p(n) + \dfrac{\beta}{d_p^-(n)}\, e_p(n) e_p^-(n)$

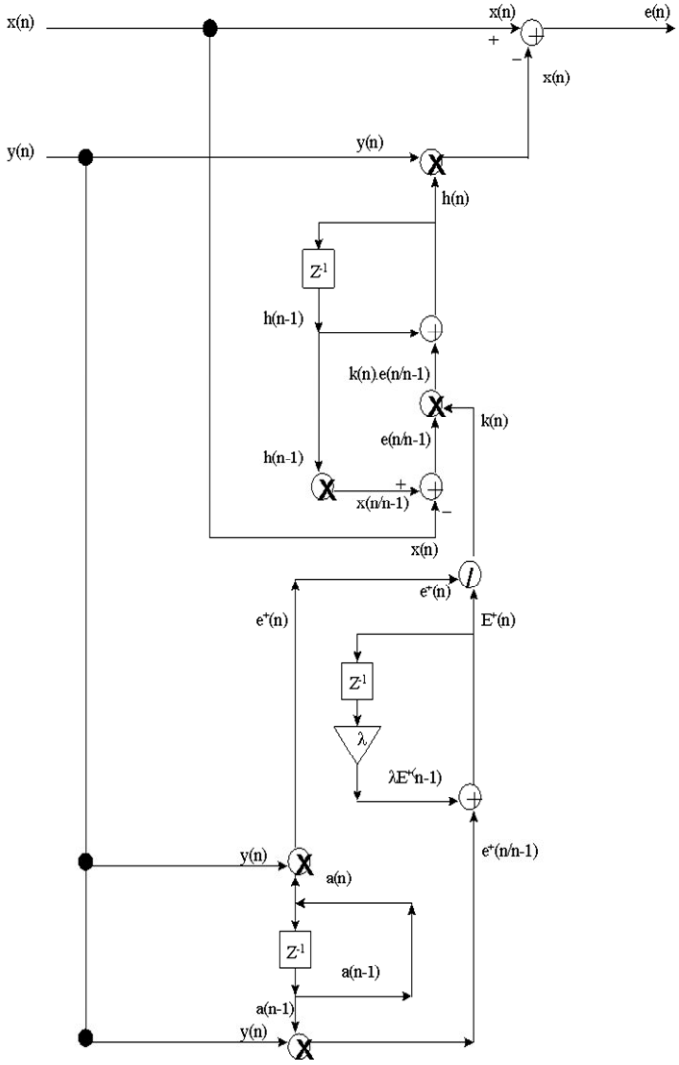
4. Teruskan untuk n selanjutnya

$\beta = \alpha/(1-\lambda)$. Biasanya harga β diambil = 1.

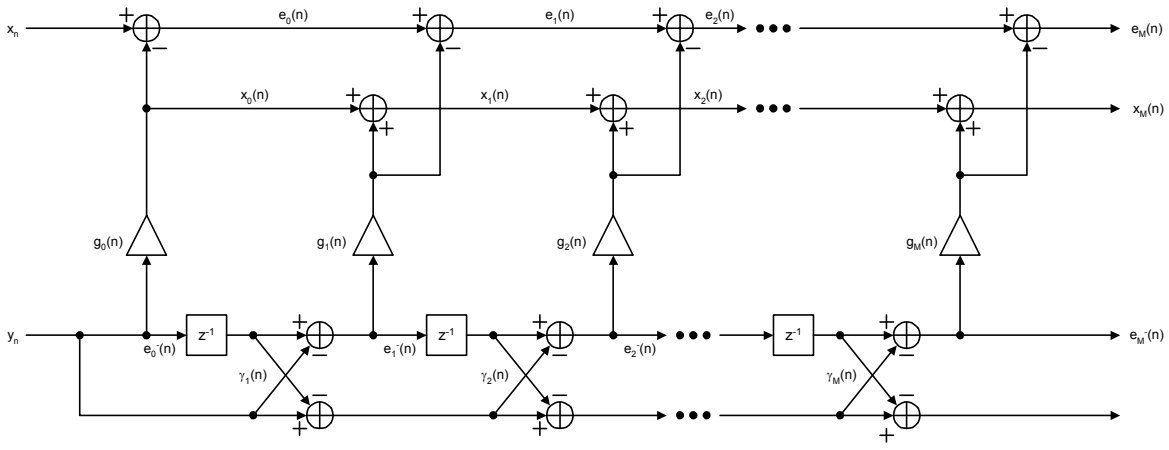
Adapun struktur filter lattice dapat dilihat pada gambar 5



Gambar 3. Struktur Filter RLS Orde 1



Gambar 4. Struktur Filter Fast Kalman untuk FIR Orde 1.



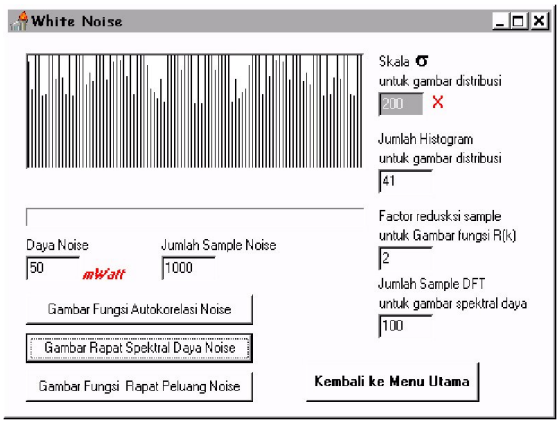
Gambar 5. Struktur Filter Latice GAL B.

4. SIMULASI

Masalah simulasi adalah bagaimana membuatnya menjadi seolah-olah dapat mewakili keadaan riil bila suatu *device* atau sistem diimplementasikan. Dalam simulasi algoritma adaptif ini ada beberapa masalah menarik yang harus diperhatikan untuk membuat simulasi ini dapat mewakili keadaan sebenarnya, yaitu:

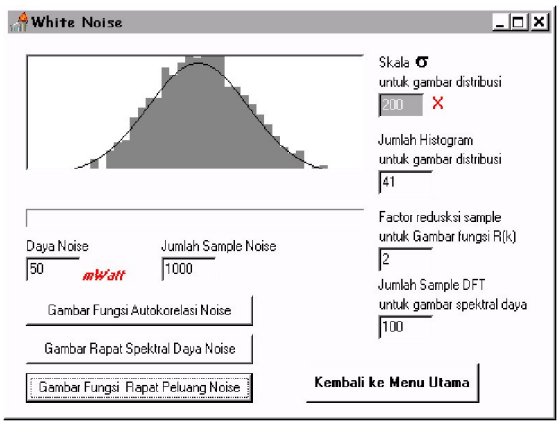
- Bahwa sinyal sinusoidal yang dibangkitkan tidak murni sinusoidal melainkan sinyal cyclo-stationer. Hal ini adalah wajar karena fungsi sinusoidal yang dipergunakan pada pemrograman ini sebenarnya adalah fungsi deret yang dibangkitkan dengan ketelitian tertentu.
- Pembangkitan *white gaussian noise* dapat dilakukan dengan menjumlahkan beberapa white noise uniform, hal ini dilakukan berdasarkan teori *central limit*.
- Pembangkitan *white noise* sendiri dilakukan dengan subrutine random yang disediakan komputer seperti dengan algoritma linear Congruential generator. Menurut spesifikasinya sebenarnya bilangan random yang dihasilkan tidak sepenuhnya random, tetapi secara praktis boleh dianggap begitu.

Pada gambar 6 terlihat spektral noise yang tidak flat ini menandakan bahwa sinyal ramdom sebenarnya tidak white noise. Ini akan berpengaruh pada simulasi karena sifat penghapus korelasi yang digunakan untuk menghapus whitenoise tidak akan 100% berhasil karena ternyata noise yang digunakan pada simulasi ini masih berkorelasi sehingga dapat muncul pada keluaran $\hat{x}_n$.



Gambar 6. Spektral White Noise

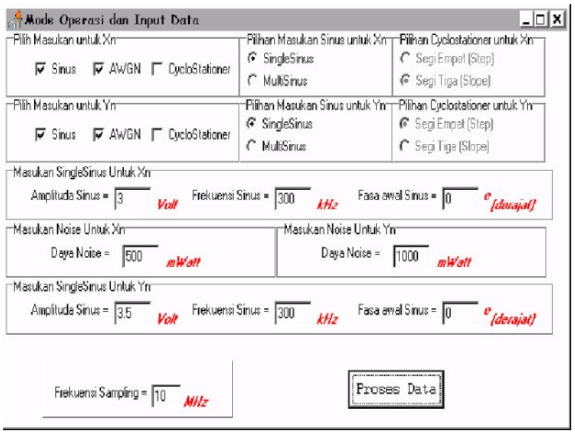
Tetapi pada pemeriksaan distribusi noisenya terlihat memang noise terdistribusi gausian seperti yang terlihat pada gambar 7.



Gambar 7. PDF Noise

4.1 Hasil Simulasi Masukan stationer (sinusoidal) untuk simulasi filter adaptif

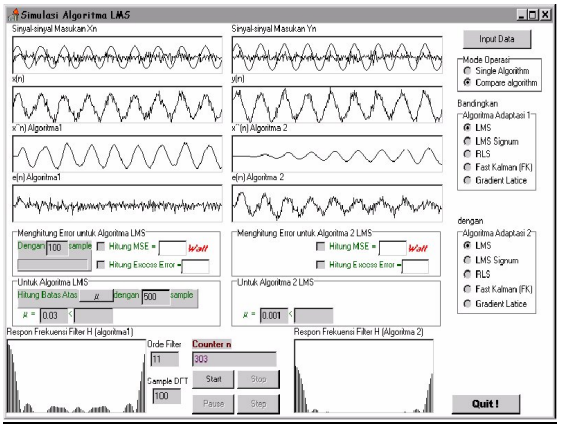
Penentuan nilai μ menentukan adanya konvergensi, kecepatan konvergensinya, dan excess errornya. Untuk menerangkan hal ini dilakukan simulasi dengan input seperti pada gambar 8 dibawah ini



Gambar 8. Masukan Stationer (Sinusoidal) Untuk Simulasi Filter Adaptif

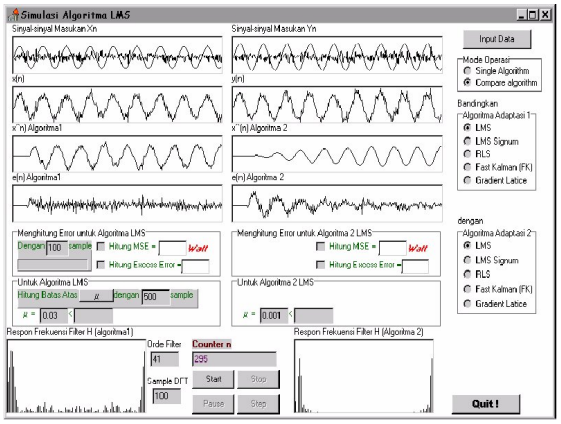
4.2 Hasil Simulasi Algoritma Least Mean Square (LMS)

Semakin kecil μ semakin teliti (*excess error* semakin kecil), tetapi semakin lama menuju konvergensi, ini dapat dilihat pada contoh hasil simulasi gambar 9 dimana diperbandingkan hasil simulasi untuk μ yang berbeda yaitu 0.03 dan 0.001



Gambar 9. Perbandingan Hasil Simulasi Algoritma LMS dengan μ = 0.03 (lebih besar) dengan μ = 0.001 (lebih kecil)

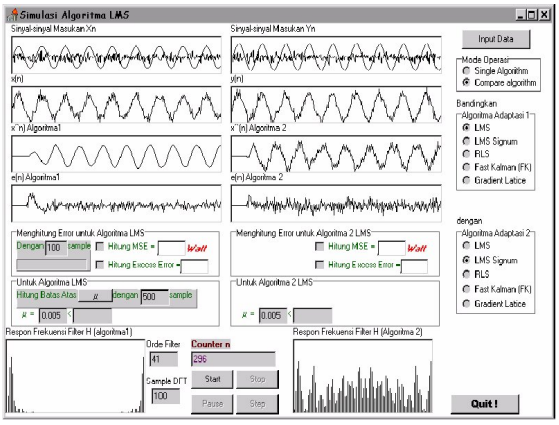
Hasil simulasi selanjutnya pada gambar 9 adalah simulasi jika orde filter diperbesar, dengan input tetap sama seperti gambar 8.



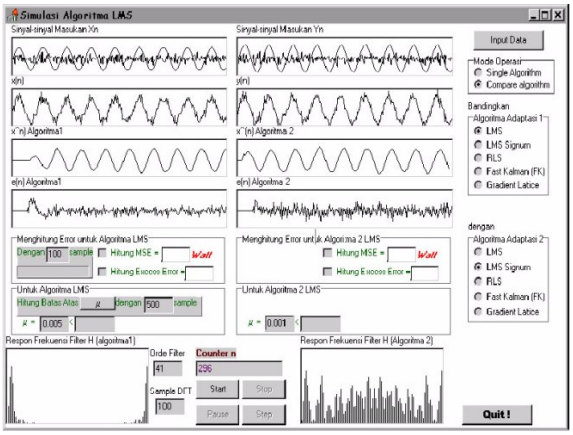
Gambar 10. Simulasi LMS dengan Orde Filter Diperbesar.

Hasil simulasi diatas menunjukkan bahwa jika orde filter diperbesar dari 11 ke 41, terlihat bahwa penambahan orde filter memperlambat waktu konvergensi, hal ini wajar karena konvergensi algoritma LMS mengikuti hubungan 20M (konvergensi dicapai setelah 20M iterasi), dimana M adalah orde filter. Meskipun konvergensi menjadi lebih lambat, ketelitian menjadi lebih besar (error lebih kecil).

Dari gambar 11 terlihat bahwa untuk μ yang sama LMS signum akan menghasilkan ketelitian (*excess error*) yang lebih jelek, tetapi hal ini dapat dikompensasi dengan memperkecil μ untuk LMS signum sehingga diperoleh kinerja yang lebih baik seperti pada gambar 12.



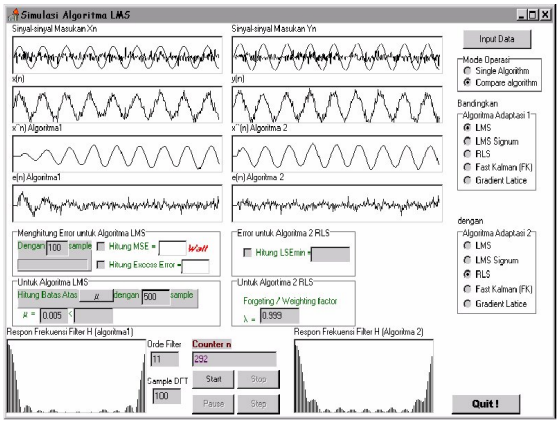
Gambar 11. Perbandingan LMS Dengan LMS Signum Dengan μ Yang Sama



Gambar 12. Perbandingan LMS dengan LMS Signum dengan μ untuk LMS Signum Jauh Lebih Kecil

4.3 Hasil Simulasi Perbandingan LMS dengan RLS

Algoritma RLS lebih cepat konvergen dari pada LMS karena lebih banyak faktor yang dipertimbangkan (algoritma RLS dapat diharapkan konvergen setelah 2M iterasi, dengan M adalah orde filter). Tetapi tentu untuk orde filter yang semakin besar maka algoritma RLS akan lebih banyak menghabiskan memori dibandingkan LMS. Hal ini dapat dilihat dari hasil simulasi berikut pada gambar 13.

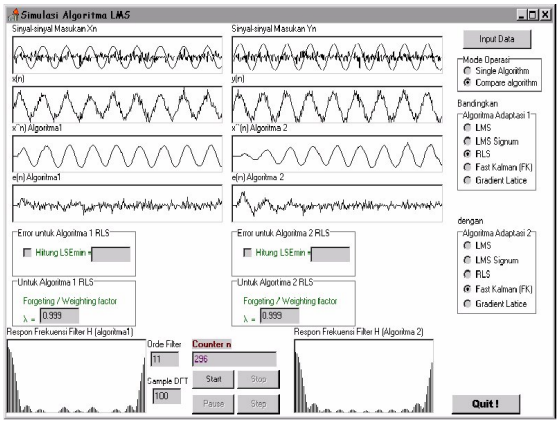


Gambar 13. Simulasi Perbandingan LMS Dengan RLS

Nilai λ pada algoritma RLS merupakan faktor pelupa untuk melupakan sample-sample sebelumnya. Untuk sinyal-sinyal stationer dibuat λ mendekati 1 (artinya sample-sample sebelumnya tidak perlu terlalu diingat). Untuk sinyal-sinyal non stationer dibuat λ mendekati 0 (artinya sample-sample sebelumnya perlu terlalu diingat).

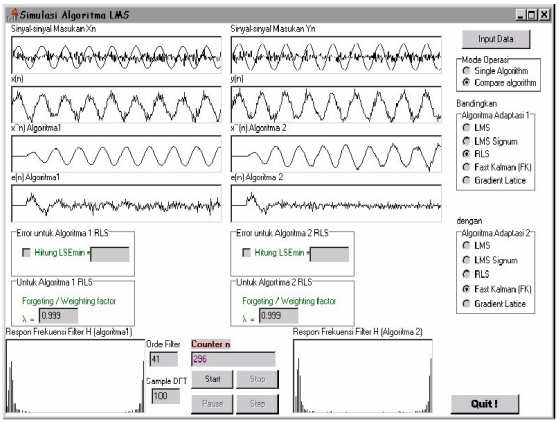
4.4 Hasil Simulasi Perbandingan Algoritma RLS dengan Fast Kalman

Walau Fast Kalman mempercepat perhitungan dibandingkan RLS, tetapi Fast Kalman tetap cepat menuju konvergensinya, contohnya dapat dilihat dari gambar 14 dan 15.



Gambar 14. Simulasi Perbandingan RLS dengan Fast Kalman dengan Orde Filter =11

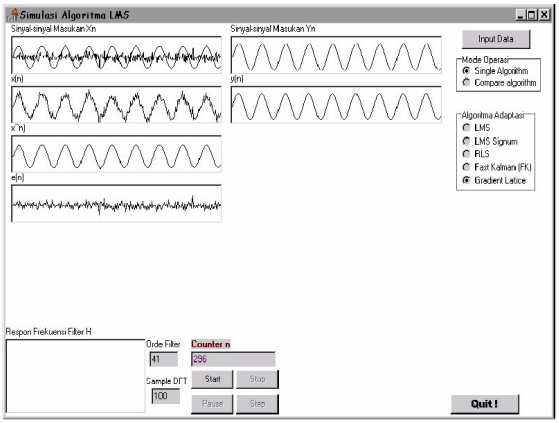
Karena banyaknya langkah algoritma yang dipergunakan, pada suatu langkah iterasi tertentu error karena proses numerik oleh proses seperti *rounding*, *truncating*, dan sebagainya akan terakumulasi, sehingga algoritma akan keluar dari kestabilannya karena masalah numerik ini, sehingga untuk mengatasi masalah ini perlu dilakukan reiinisialisasi. Reinisialisasi ini tidak perlu diterapkan kepada seluruh variabel, tetapi cukup variabel **k**(n) yang diinisialisasi. Hal tersebut disimulasikan pada gambar 11, dimana untuk orde filter lebih tinggi (41) filter Fast Kalman akan menuju ke kinerja yang lebih buruk setelah beberapa sampel, sebelum direinisialisasi lagi. Hal ini wajar karena untuk orde filter yang lebih besar berarti perhitungan juga lebih besar, sehingga akumulasi error juga lebih cepat.



Gambar 15. Simulasi perbandingan RLS dengan Fast Kalman dengan orde filter = 41

4.5 Hasil Simulasi Algoritma Gradient Adaptive Lattice

Hasil simulasinya dapat dilihat pada gambar 16



Gambar 16. Simulasi Gradient Adaptive Lattice

5. ANALISA DAN KESIMPULAN

Untuk menghasilkan simulasi yang baik ada beberapa hal yang harus diantisipasi dan dibuat agar simulasi dapat benar-benar mewakili kejadian sebenarnya, seperti masalah

pembangkitan sinyal, pengambilan sampelnya, dan kausalitas dari simulasi.

Algoritma adaptif sendiri terdiri dari bermacam-macam jenisnya, tetapi semuanya berangkat dari dasar filter optimum yang sama, perbedaannya terutama adalah dari masalah ketelitian, kecepatan konvergensinya, dan kestabilan numerik simulasinya.

Algoritma adaptif sendiri dapat dibagi tiga secara garis besar perkembangannya yaitu yang berdasarkan LMS, RLS dan dibuatnya struktur lattice untuk kedua algoritma tersebut.

Perilaku masing-masing algoritma yang diamati adalah sebagai berikut:

1. **Algoritma LMS (*Least Mean Square)*** Algoritma ini sederhana dan stabil, akan tetapi memiliki kecepatan konvergensi yang sangat lambat. Ketelitian algoritma LMS dapat ditingkatkan dengan menaikkan jumlah orde filter atau memperkecil μ. Akan tetapi kedua cara ini memperlambat konvergensi filter. Varian algoritma LMS, algoritma LMS Signum, memiliki kinerja yang lebih jelek dibandingkan algoritma LMS, namun lebih sederhana.
2. **Algoritma RLS *(Recursive Least Square)* dan Fast Kalman**. Kedua algoritma ini memiliki kecepatan konvergensi yang tinggi, akan tetapi lebih kompleks dan memiliki kestabilan yang kurang baik, sehingga harus dilakukan reinisialisasi pada variabel **k** untuk menjaga stabilitas algoritma.
3. **Algoritma GAL (*Gradient Adaptive Lattice*)** Algoritma ini memiliki kecepatan konvergensi yang sangat tinggi, dan juga memiliki kestabilan numerik. Algoritma ini merupakan algoritma yang paling kompleks dari algoritma yang disimulasikan.
4. Ada hal yang harus sangat diperhatikan dalam simulasi atau realisasi algoritma adaptasi dalam hardware nantinya, yaitu masalah error karena proses-proses numerik seperti *rounding, truncating*, dan sebagainya, yang jika terakumulasi akan menimbulkan masalah, sehingga harus sangat diperhatikan masalah banyaknya bit yang menyatakan suatu nilai, error karena penggunaan deret atas perhitungan tertentu seperti sinus, dan lain-lain.
5. Hal lain yang dapat dicatat adalah bahwa dalam melakukan simulasi, yang diimplementasikan adalah suatu sistem yang tidak real-time. Dalam percobaan ini tampak jelas bahwa yang terjadi adalah cuplikan sinyal datang "menunggu" perhitungan filter selesai.Hal ini tidak dapat terjadi pada sistem *real-time.* Oleh karena itu, dalam realisasi algoritma dalam sistem *real-time*, harus diperhitungkan waktu yang dibutuhkan untuk melaksanakan perhitungan sehingga perhitungan dapat selesai sebelum cuplikan sinyal berikutnya tiba.

6. DAFTAR PUSTAKA

[1] Orfanidis, Sophocles J. *Optimum Signal Processing, An Introduction*, 2nd ed., McGraw-Hill Book Company, 1988.

[2] LIN, David W., *On Digital Implementation of the Fast Kalman Algorithms*, IEEE Transaction on Acoustics, Speech, and Signal Processing., vol. ASSP-32, hal. 998-1005, No.5, October 1984.

[3] Hayes, Monson H., *Statistical Digital Signal Processing and Modeling*, John Wiley & Sons, Inc., 1996.

[4] Haykin, Simon, *Introduction to Adaptive Filters*, MacMillan Publishing Company, 1984.

[5] Haykin, Simon, *Adaptive Filter Theory*, Prentice-Hall International Inc., 1996.