bidang
TEKNIK

STUDI KOMPARASI ALGORITMA *PARTICLE SWARM OPTIMIZATION* PADA APLIKASI *FILTER ADAPTIVE NOISE CANCELLATION*

TRI RAHAJOENINGROEM
MUHAMMAD ARIA

Jurusan Teknik Elektro– FTIK
Universitas Komputer Indonesia

Adaptive Noise Cancellation (ANC) adalah aplikasi filter adaptif yang digunakan untuk mengatasi sinyal yang terdistorsi dengan *noise*. Menggunakan ANC diharapkan dapat direkonstruksi ulang sinyal asli dengan *noise* yang minimal. Filter adaptif adalah filter yang dapat menyesuaikan koefisien-koefisiennya terhadap perubahan sistem. Pada penelitian ini akan diuji penerapan beberapa algoritma *Particle Swarm Optimization* (PSO) pada algoritma filter adaptif. Algoritma PSO yang akan dibandingkan adalah Original PSO, Local PSO, Canonical PSO, Decreasing Inertia Weight PSO, Increasing Inertia Weight PSO, Stochastic Inertia Weight PSO, Fully Informed PSO, Self-Organizing Hierarchical PSO with Time-Varying Acceleration Coefficients, Hierarchical PSO, Adaptive Hierarchical PSO dan Estimation of Distribution PSO. Untuk mengukur kinerja dari filter adaptif tersebut maka akan digunakan *mean square error* (MSE). Algoritma PSO yang menghasilkan rata-rata MSE terbaik adalah Estimation of Distribution PSO dengan MSE sebesar 3,87 x 10^{-2}.

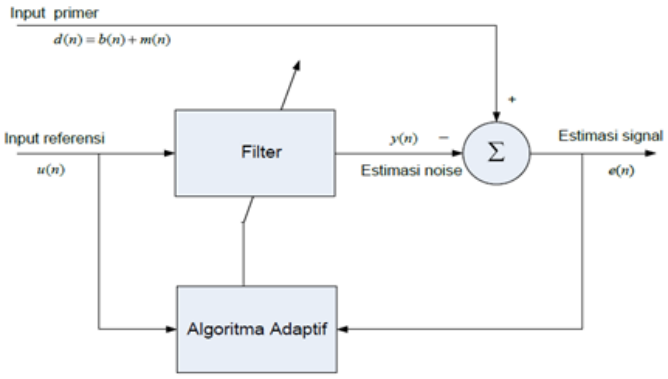
Kata kunci – Adaptive Noise Cancellation, Algoritma Particle Swarm Optimization, Filter Adaptif, Mean Square Error

PENDAHULUAN

Dalam beberapa sistem telekomunikasi, suatu sinyal yang diterima dapat dirusak oleh *noise* (gangguan) seperti suara manusia yang terlemahkan oleh suara percakapan orang lain. Sehingga sinyal yang diterima adalah sinyal asli yang telah tercampur dengan *noise*. Penggunaan filter tapis bawah (*lowpass filter*), filter tapis atas (*highpass filter*) maupun filter tapis tengah (*bandpass filter*) memang dapat digunakan walaupun terkadang hasil yang diperoleh tidak maksimal. Maka pada penelitian ini akan diterapkan metode alternatif menggunakan filter *Adaptive Noise Cancellation* (ANC).

Filter adaptif adalah filter dapat mengubah bobot koefisiennya secara otomatis, menyesuaikan dengan kesalahan *(error*) yang terjadi. Pada aplikasi *ANC*, filter adaptif bertujuan untuk menghilangkan *noise* sehingga akan mendapatkan error yang terkecil dari sinyal berdasarkan pengukuran *Mean Square Error* (MSE) dan *Signal to Noise Ratio* (SNR). Salah satu metode algoritma filter adaptif yang sering digunakan adalah algoritma *Least Mean Square* (LMS).

Particle Swarm Optimization (PSO) adalah salah satu algoritma optimasi berbasis kecerdasan buatan terdistribusi yang diinspirasikan oleh kecerdasan kumpulan burung dan ikan. PSO pertama kali diperkenalkan oleh Kennedy dan Eberhart [1]. Selain PSO, algoritma kecerdasan buatan terdistribusi lainnya adalah *Genetic Algorithm (GA)* dan *Bee Colony Optimization (BCO).* Telah banyak peneliti yang mencoba mengimplementasikan algoritma



Gambar 1. Diagram blok ANC

kecerdasan buatan terdistribusi ini pada kasus *Adaptive Noise Cancellation*. Karaboga [2] mengimplementasikan BCO pada ANC, Renaldi [3] mengimplementasikan GA pada ANC dan Ben Arfia [4] mengimplementasikan PSO pada Filter adaptif. Tetapi sepanjang yang penulis ketahui, belum ada penelitian yang membandingkan beberapa algoritma PSO untuk diimplementasikan pada kasus ANC ini.

Pada penelitian ini akan diuji penerapan beberapa algoritma *Particle Swarm Optimization* (PSO) pada algoritma filter adaptif. Algoritma PSO yang akan dibandingkan adalah Original PSO, Local PSO, Canonical PSO, Decreasing Inertia Weight PSO, Increasing Inertia Weight PSO, Stochastic Inertia Weight PSO, Fully Informed PSO, Self-Organizing Hierarchical PSO with Time-Varying Acceleration Coefficients, Hierarchical PSO, Adaptive Hierarchical PSO dan Estimation of Distribution PSO. Sebagai ukuran kinerja, akan digunakan *mean square error* (MSE).
Susunan makalah ini yaitu pada bagian 2, dijelaskan mengenai teori Adaptive Noise Cancellation (ANC). Pada bagian 3, dibahas algoritma-algoritma *Particle Swarm Optimization* yang digunakan. Bagian 4 menyajikan hasil implementasi dan pengujian data, dan bagian 5 adalah kesimpulan.

TEORI DASAR

Adaptive Noise Cancellation (ANC)

ANC pertama kali diperkenalkan oleh Bernard Widrow [5] pada tahun 1975. Aplikasi dari ANC ditunjukkan pada blok diagram seperti **Gambar 1** berikut. Tujuan dari ANC adalah membuang noise m(n) dari sinyal primer d(n) sehingga akan diperoleh kembali rekonstruksi dari sinyal informasi b (n).

ANC memerlukan dua input, yaitu :

1. Input primer, d(n) adalah sinyal pembawa informasi b(n) yang terdistorsi dengan sinyal noise m(n). Sinyal b(n) dan m(n) tidak berkorelasi antara satu dengan lainnya
2. Input referensi adalah sebuah sinyal noise u(n) yang berkorelasi dengan m(n) tetapi tidak berkorelasi dengan b(n). Perlu ditekankan bahwa jika u(n) tidak berkorelasi dengan m(n) maka tidak mungkin mengoptimasi b(n)

Sinyal referensi u(n) diolah oleh filter adaptif melalui persamaan

$$y(n) = \sum_{k=0}^{M-1} w_k(n) \cdot u(n-k) \qquad (1)$$

Dimana $w_k(n)$ adalah koefisien filter yang dapat diatur. Sinyal primer d(n) akan dikurangkan dengan keluaran filter y(n)

sehingga menjadi respon yang diinginkan untuk filter adaptif. Sinyal error didefinisikan sebagai :

$$e(n) = d(n) - y(n) \quad (2)$$

Karena d(n) = b(n) + m(n), maka :

$$e(n) = b(n) + m(n) - y(n) \quad (3)$$

Sinyal error e(n) akan digunakan untuk menyesuaikan nilai koefisien filter. Filter adaptif akan berusaha untuk meminimalkan nilai kuadrat rata-rata dari sinyal error e(n).

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) adalah salah satu algoritma optimasi berbasis kecerdasan buatan terdistribusi yang diinspirasikan oleh kecerdasan kumpulan burung dan ikan. Pergerakan partikel akan ditentukan oleh nilai posisi saat ini dan nilai kecepatan. Nilai posisi dari partikel akan merepresentasikan solusi yang mungkin pada kasus optimasi, sedangkan nilai kecepatan digunakan untuk merubah posisi partikel. Terdapat beberapa algoritma PSO yang dikembangkan oleh para peneliti. Algoritma-algoritma yang akan diteliti pada penelitian ini ditunjukkan pada **Tabel 1** .

The Original Particle Swarm Optimization Algorithm (Original PSO)

Algoritma Original PSO diperkenalkan oleh J. Kennedy [6] pada tahun 1995. Algoritma Original PSI dimulai dengan inisialisasi populasi solusi (disebut particle) yang dibangkitkan secara acak. Nilai dari fungsi objektif partikel-partikel tersebut akan mendeskripsikan kualitas dari posisi partikel atau alternatif solusi tersebut.

Setiap particel p_i pada waktu *t* akan memiliki vektor posisi $\vec{x}_i^t$ dan vektor kecepatan $\vec{v}_i^t$. Setiap particle juga akan mencatat posisi terbaik yang pernah dicapainya. Variabel yang digunakan untuk mencatat posisi terbaik yang pernah dicapai oleh suatu partikel biasanya menggunakan vektor $pb\vec{e}st_i$. Nilai dari vektor ini akan diperbaharui setiap kali partikel yang bersangkutan memperoleh solusi yang lebih baik. Adapun vektor $gb\vec{e}st$ merepresentasikan posisi terbaik yang pernah dicapai oleh suatu partikel dari seluruh populasi.

Algoritma PSO akan beriterasi memperbaharui nilai posisi dan kecepatan partikel higga kondisi berhenti tercapai.

Tabel 1.
Beberapa algoritma PSO yang diteliti

Algoritma	Sumber pustaka
Original PSO	J. Kennedy, et al. (1995)
Local PSO	R. Eberhart, et al. (1995)
Canonical PSO	M. Clerc, et al. (2002)
Decreasing Inertia Weight PSO	Y. Shi, et al. (1999)
Increasing Inertia Weight PSO	Y.L. Zheng, et al. (2003)
Stochastic Inertia Weight PSO	R. Eberhart, et al. (2001)
Fully Informed PSO	R. Mendes, et. Al. (2004)
Self-Organizing Hierarchical PSO with Time-Varying Acceleration Coefficients	A. Ratnaweera, et al. (2004)
Hierarchical PSO	C.-C. Chen (2009)
Adaptive Hierarchical PSO	S. Janson, et al. (2005)
Estimation of Ditribution PSO	M. Iqbal, et al

Persamaan *update rule* yang digunakan adalah :

$$\vec{v}_i^{t+1} = \vec{v}_i^{t} + \varphi_1 \cdot \vec{U}_1(0,1) * (pb\vec{e}st_i - \vec{x}_i^{t}) + \varphi_2 \cdot \vec{U}_2(0,1) * (gb\vec{e}st - \vec{x}_i^{t}) \quad (4)$$

$$\vec{x}_i^{t+1} = \vec{x}_i^{t} + \vec{v}_i^{t+1} \quad (5)$$

Dimana φ_1 dan φ_2 adalah dua nilai konstan yang disebut *cognitive* dan *social acceleration coefficients*, $\vec{U}_1(0,1)$ dan $\vec{U}_2(0,1)$ adalah vektor dua dimensi yang memiliki nilai acak yang terdistribusi merata yang dibangkitkan setiap kali iterasi dimana nilainya akan berada diantara rentang nilai 0 dan 1. Sedangkan * adalah operator perkalian vektor elemen per elemen.

Pada algoritma *original PSO*, setiap partikel memiliki dua penarik pergerakan yaitu $pb\vec{e}st_i$ dan $gb\vec{e}st$. Tetapi berdasarkan beberapa penelitian menunjukkan bahwa penarikan yang terlalu kuat ke posisi $gb\vec{e}st$ akan mengakibatkan konvergensi yang terlalu cepat. Maka diusulkanlah variasi dari *original PSO* yang disebut sebagai *local PSO*.

Local Particle Swarm Optimizer (Local PSO)

Algoritma *local PSO* diperkenalkan oleh R. Eberhart [7] pada tahun 1995. Prinsipnya adalah bahwa suatu partikel tidak mengalami percepatan yang diakibatkan oleh $gb\vec{e}st$, tetapi akan mengalami percepatan yang diakibatkan oleh posisi terbaik yang pernah dicapai oleh partikel-partikel yang menjadi tetangga partikel bersangkutan (bukan posisi terbaik dari seluruh partikel). Maka, persamaan (4) akan menjadi

$$\vec{v}_i^{t+1} = \vec{v}_i^{t} + \varphi_1 \cdot \vec{U}_1(0,1) * (pb\vec{e}st_i - \vec{x}_i^{t}) + \varphi_2 \cdot \vec{U}_2(0,1) * (lb\vec{e}st_i - \vec{x}_i^{t}) \quad (6)$$

Dimana $lb\vec{e}st_i$ adalah posisi terbaik yang pernah dicapai oleh tetangga-tetangga dari suatu partikel. Mohais [8] melaporkan bahwa pemilihan ketertetanggaan partikel secara acak menghasilkan performansi yang sama atau terkadang lebih baik daripada pemilihan ketertetanggaan partikel menggunakan topologi.

Canonical Particle Swarm Optimizer

Clerc dan Kennedy [9] pada tahun 2002 memperkenalkan penggunaan *constriction factor* pada *update rule* kecepatan. Tujuan penggunaan *constriction factor* adalah mencegah penambahan kecepatan partikel menuju nilai yang terlalu besar dan juga untuk mengendalikan konvergensi dari partikel.

Variabel *constriction factor* ini ditambahkan pada persamaan (4) sehingga menjadi

$$\vec{v}_i^{t+1} = \mathcal{X} \cdot (\vec{v}_i^{t} + \varphi_1 \cdot \vec{U}_1(0,1) * (pb\vec{e}st_i - \vec{x}_i^{t}) + \varphi_2 \cdot \vec{U}_2(0,1) * (lb\vec{e}st_i - \vec{x}_i^{t})) \quad (7)$$

dengan

$$\mathcal{X} = \frac{2 \cdot k}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad (8)$$

Dengan $k \in [0,1]$, $\varphi = \varphi_1 + \varphi_2$ dan $\varphi > 4$.

Time-Varying Decreasing Inertia Weight Particle Swarm Optimizer (Decreasing Inertia Weight PSO

Shi dan Eberhart [10] pada tahun 1999 memperkenalkan ide penambahan

pengontrol yang disebut sebagai *inertia weight* untuk mengontrol diversifikasi dari *original PSO*. Maka persamaan (4) akan menjadi

$$\vec{v}_i^{t+1} = w(t) \cdot \vec{v}_i^{t} + \varphi_1 \cdot \vec{U}_1(0,1) * \left(\vec{pbest}_i - \vec{x}_i^{t}\right) + \varphi_2 \cdot \vec{U}_2(0,1) * \left(\vec{lbest}_i - \vec{x}_i^{t}\right) \quad (9)$$

dengan *w(t)* adalah *inertia weight* yang biasanya memiliki nilai yang tergantung dengan iterasi. Biasanya φ_1 dan φ_2 diset bernilai 2.

Persamaan *inertia weight* yang digunakan adalah

$$w(t) = w_{max} - \frac{(w_{max} - w_{min}) \cdot t}{t_{max}} \quad (10)$$

dimana t_{max} adalah waktu saat nilai *w(t)* = w_{min}. *Variabel* w_{max} dan w_{min} adalah nilai maksimum dan minimum dari *inertia weight* yang dikehendaki.

Time-Varying Increasing Inertia Weight Particle Swarm Optimizer (Increasing Inertia Weight PSO

Zheng [11] pada tahun 2003 meneliti bahwa efek penambahan nilai *inertia weight*, pada beberapa kasus, menjadikan hasil konvergensi yang lebih baik. Maka persamaan *inertia weight* yang digunakan adalah

$$w(t) = w_{min} - \frac{(w_{min} - w_{max}) \cdot t}{t_{max}} \quad (11)$$

Time-Varying Stochastic Inertia Weight Particle Swarm Optimizer (Stochastic Inertia Weight PSO)

Eberhart dan Shi [12] pada tahun 2001 mengajukan variasi lainnya dari penggunaan *inertia weight* yaitu dengan cara penentuan nilai *inertia weight* tersebut sacara acak dalam rentang 0.5 dan 10.

Fully Informed Particle Swarm Optimizer (Fully Informed PSO)

Mendes [13] pada tahun 2004 mengusulkan algoritma yang bernama *Fully Informed PSO*, dimana suatu partikel akan menggunakan informasi mengenai posisi terbaik yang pernah dicapai oleh setiap partikel.

Maka *update rule* yang digunakan adalah

$$\vec{v}_i^{t+1} = \chi \left[\vec{v}_i^{t} + \sum_{k \in \mathcal{N}_i} \varphi_k \cdot W(\vec{pbest}_k) \cdot \vec{U}_k(0,1) * \left(\vec{pbest}_k - \vec{x}_i^{t}\right) \right] \quad (12)$$

dimana $\mathcal{N}_i$ adalah tetangga dari partikel *i*, $W(\vec{pbest}_k)$ adalah fungsi pembobotan. Tujuannya dari $W(\vec{pbest}_k)$ adalah memberikan informasi mengenai kualitas pengaruh $\vec{pbest}_i$.

Self Organizing Hierarchical Particle Swarm Optimizer with Time-varying Acceleration Coefficients (HPSOTYAC)

HPSOTVAC diusulkan oleh Ratnaweera [14] pada tahun 2004. Jika komponen kecepatan suatu partikel menjadi nol, maka kecepatannya akan diset ulang menjadi kecepatan maksimalnya. Nilai koefisien percepatan juga akan diadaptasikan sesuai dengan iterasi. *Cognitive coefficient* akan berkurang dari nilai 2,5 menuju 0,5 sedangkan nilai *social coefficient* akan bertambah dari 0,5 menuju 2,5.

Nilai kecepatan maksimal pada HPSOTVAC juga akan diadaptasikan dari V_{max} menjadi 0,1V_{max}.

Hierarchical Particle Swarm Optimizer (Hierarchical PSO)

H-PSO diusulkan oleh Chen [15] pada tahun 2009. Pada H-PSO seluruh partikel akan disusun berdasarkan suatu hirarki. Setiap partikel akan bertetangga dengan

partikel yang berada pada hierarki diatasnya (disebut *parent*).

Hirarki partikel akan didefinisikan oleh *height h*, *branching degree d*, jumlah maksimum *children* pada setiap node dan jumlah node *m* yang akan ada pada hirarki. Pada HPSO digunakan hirarki yang memiliki jumlah *children* yang sama untuk setiap node-nya. Hanya node-node terbawah saja yang boleh memiliki jumlah *children* yang lebih sedikit.

Posisi baru dari suatu partikel akan ditentukan berdasarkan posisi terbaik dari partikel tersebut dan posisi terbaik dari *parent*. Jika suatu partikel mencapai nilai yang lebih baik dari *parent*, maka partikel dan *parent* partikel tersebut akan bertukar posisi pada hirarki.

Adaptive Hierarchical Particle Swarm Optimizer (AH-PSO)

Diusulkan oleh Jansen dan Middendorf [16] pada tahun 2005. Pada *AH-PSO, branching degree* dari hirarki akan dikurangkan sedikit demi sedikit selama iterasi berlangsung. *Branching degree* akan dikurangkan berdasarkan variabel k_{adapt} sehingga nilai minimum d_{min} tercapai. Pengurangan akan dilakukan setiap f_{adapt}.

Estimation of Distribution Particle Swarm Optimizer (EDPSO)

Diusulkan oleh Iqbal [17], EDPSO meminjam beberapa ide dari *Continues ACO*. EDPSO akan bekerja seperti *Caconical PSO* dengan beberapa modifikasi. Setiap kali *update rule* dieksekusi, maka EDPSO akan memilih suatu fungsi Gaussian. Kemudian dari fungsi Gaussian tersebut akan dievaluasi kemungkinan partikel untuk pindah ke posisi baru tersebut. Jika aturan perpindahan partikel terpenuhi, maka partikel akan berpindah seperti biasa, tetapi jika aturan perpindahan tidak terpenuhi, maka akan dilakukan sampling dari fungsi Gaussian tersebut seperti pada *Continues ACO*.

IMPLEMENTASI DAN ANALISA DATA

Untuk pengujian, akan digunakan parameter-parameter PSO seperti yang ditunjukkan pada **Tabel 2** berikut.

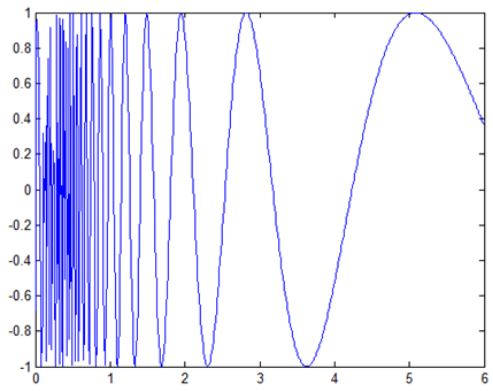
Table 2. Seting parameter PSO yang digunakan

Algoritma	Seting
Original PSO	Cognitive component (φ_1) = 2.05 Social component (φ_2) = 2.05
Local PSO	Cognitive component (φ_1) = 2.05 Social component (φ_2) = 2.05 Number of Neighborhood = 3
Canonical PSO	Cognitive component (φ_1) = 2.05 Social component (φ_2) = 2.05 Constriction factor (χ) = 0.729 Number of Neighborhood = 3
Decreasing Inertia Weight	Cognitive component (φ_1) = 2.05 Social component (φ_2) = 2.05 Initial inertia weight = 0.9 Final inertia weight = 0.4 Number of Neighborhood = 3
Increasing Inertia Weight	Cognitive component (φ_1) = 2.05 Social component (φ_2) = 2.05 Final inertia weight = 0.9 Initial inertia weight = 0.4 Number of Neighborhood = 3
Stochastic Inertia Weight	Cognitive component (φ_1) = 2.05 Social component (φ_2) = 2.05 Minimum inertia weight = 0.4 Maximum inertia weight = 0.9 Number of Neighborhood = 3

Table 2. Seting parameter PSO yang digunakan (Lanjutan)

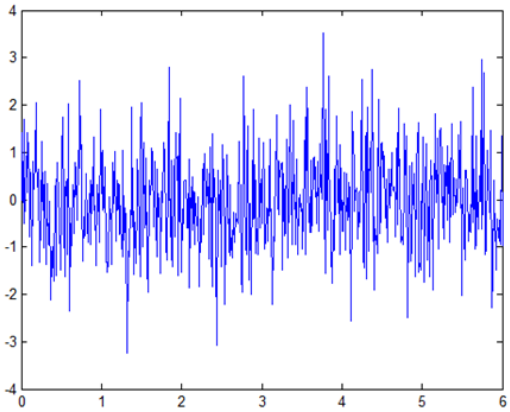
Algoritma	Seting
Fully Informed PSO	Sum of the acc. coeff. (φ) = 4.1 Constriction factor (χ) = 0.729 Number of Neighborhood = 3
Self Organizing Hierarchical PSO	Initial value of (φ_1) = 2.5 Final value of (φ_1) = 0.5 Initial value of (φ_2) = 0.5 Final value of (φ_2) = 0.5 Number of Neighborhood = 3
Hierarchical PSO	Cognitive component (φ_1) = 2 Social component (φ_2) = 2 w = 0.9 r = 0.95 height of the tree (h) = 3 branching degree (d) = 4
Adaptive Hierarchical PSO	Cognitive component (φ_1) = 2.05 Social component (φ_2) = 2.05 Constriction factor (χ) = 0.729 Initial Branching factor = 20 d min = 2, f adapt/m = 10 k adapt = 3
Estimation of Distribution PSO	Cognitive component (φ_1) = 2.05 Social component (φ_2) = 2.05 Constriction factor (χ) = 0.729 q = 0.1, epsilone = 0.85

Sinyal informasi b(n) menggunakan gelombang sinusoidal seperti pada **Gambar 2** berikut



Gambar 2. Sinyal informasi b(n)

Sinyal gangguan u(n) dibangkitkan secara acak seperti pada **Gambar 3** berikut.

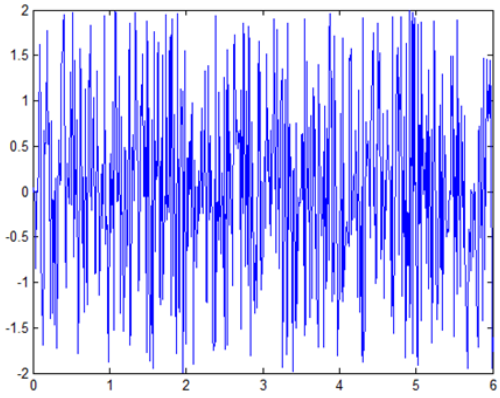


Gambar 3. Sinyal gangguan u(n)

Korelasi u(n) dengan m(n) adalah sebagai berikut.

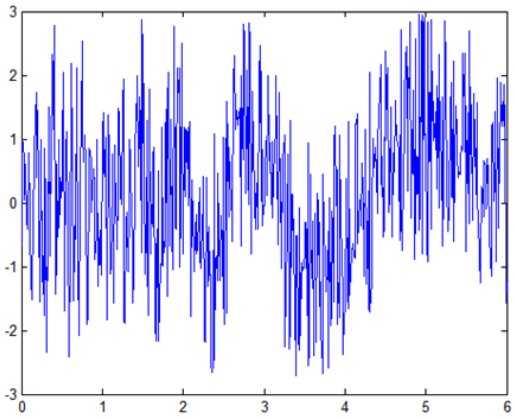
$$m(n) = \frac{4\sin(u(n)) \cdot u(n-1)}{(1+u(n-1)^2)} \qquad (13)$$

Sehingga sinyal gangguan m(n) yang diperoleh adalah seperti **Gambar 4** berikut.



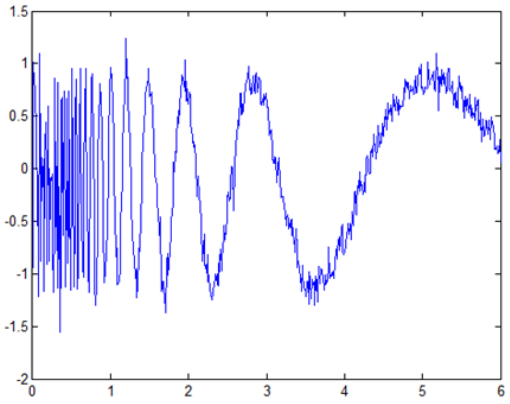
Gambar 4. Sinyal gangguan m(n)

Sinyal pengukuran d(n) adalah hasil interferensi sinyal b(n) dan m(n) yang ditunjukkan pada **Gambar 5** berikut.



Gambar 5. Sinyal informasi dan *noise*

Tujuan dari ANC adalah memperoleh kembali sinyal b(n), seperti yang ditunjukkan pada **Gambar 2,** dari sinyal pengukuran d (n), seperti yang ditunjukkan pada **Gambar 5**. Salah satu contoh hasil implementasi PSO pada ANC menggunakan algoritma *Original PSO* ditunjukkan pada **Gambar 6** berikut.



Gambar 6. Hasil implementasi PSO pada ANC menggunakan algoritma *Original PSO*

Adapun hasil pengujian lengkap dari setiap algoritma PSO ditunjukkan pada **Tabel 3** berikut

KESIMPULAN

Berdasarkan hasil implementasi, dapat diambil kesimpulan sebagai berikut.

1. Beberapa pengembangan algoritma PSO, yaitu *Original PSO, Local PSO, Canonical PSO, Decreasing Inertia Weight PSO, Increasing Inertia Weight PSO, Stochastic Inertia Weight PSO, Fully Informed PSO, Self -Organizing Hierarchical PSO with Time-Varying Acceleration Coefficients, Hierarchical PSO, Adaptive Hierarchical PSO* dan *Estimation of Distribution PSO* telah berhasil diimplementasikan pada kasus *Adaptive Noise Cancellation*.
2. Algoritma PSO yang menghasilkan rata-rata MSE terbaik adalah *Estimation of Distribution PSO* dengan MSE sebesar 3,87 x 10^{-2}.

Tabel 3 Hasil pengujian implementasi beberapa algoritma PSO pada kasus ANC

Algoritma	MSE	
Original PSO	Mean	$6{,}60 \times 10^{-2}$
	Min	$2{,}04 \times 10^{-2}$
	Max	$1{,}60 \times 10^{-1}$
Local PSO	Mean	$6{,}05 \times 10^{-2}$
	Min	$2{,}03 \times 10^{-2}$
	Max	$2{,}28 \times 10^{-1}$
Canonical PSO	Mean	$1{,}13 \times 10^{-1}$
	Min	$2{,}03 \times 10^{-2}$
	Max	$3{,}57 \times 10^{-1}$
Decreasing Inertia Weight	Mean	$6{,}55 \times 10^{-2}$
	Min	$2{,}04 \times 10^{-2}$
	Max	$2{,}47 \times 10^{-1}$
Increasing Inertia Weight	Mean	$4{,}28 \times 10^{-2}$
	Min	$2{,}15 \times 10^{-2}$
	Max	$1{,}08 \times 10^{-1}$
Stochastic Inertia Weight	Mean	$5{,}37 \times 10^{-2}$
	Min	$2{,}03 \times 10^{-2}$
	Max	$1{,}24 \times 10^{-1}$
Fully Informed PSO	Mean	$7{,}09 \times 10^{-2}$
	Min	$2{,}03 \times 10^{-2}$
	Max	$1{,}87 \times 10^{-1}$
Self Organizaing Hierarchical PSO	Mean	$9{,}17 \times 10^{-2}$
	Min	$2{,}07 \times 10^{-2}$
	Max	$2{,}81 \times 10^{-1}$
Hierarchical PSO	Mean	$5{,}15 \times 10^{-2}$
	Min	$4{,}11 \times 10^{-2}$
	Max	$9{,}41 \times 10^{-2}$
Adaptive Hierarchical PSO	Mean	$7{,}93 \times 10^{-2}$
	Min	$2{,}07 \times 10^{-2}$
	Max	$3{,}39 \times 10^{-1}$
Estimation of Distribution PSO	Mean	$3{,}87 \times 10^{-2}$
	Min	$2{,}05 \times 10^{-2}$
	Max	$1{,}04 \times 10^{-1}$

DAFTAR PUSTAKA

J. Kennedy and R. Eberhart (1995). *"Particle Swarm Optimization"*, Proc. IEE Int. Conf Neural Networks (ICNN'95), vol. IV, Perth, Australia, pp 1942 – 1948

Nurhan Karaboga, (2011), *"A Novel and Efficient Algorithm for Adaptive Filtering : Artificial Bee Colony Algorithm"*, Turk J Elec Eng & Comp Sci, Vol. 19, No. 1, 2011, pp 175 – 190

Renaldi B., Sumardi, Sudjadi, *Perbandingan Kinerja Algoritma LMS dan Algoritma Genetik Untuk Filter Adaptif Penghilang Noise*

Faten Ben Arfia, (2009), *"Nonlinear Adaptive Filters Based on Particle Swarm Optimization"*, Leonardo Journal of Sciences, Issues 14, January – June 2009, pp 244 – 251

Bernard Widrow, et all, *Adaptive Noise Cancelling : Principles and Applications*, Proceedings of The IEEE, Vol. 63, No. 12, December 1975, pp 1692 - 1707

J. Kennedy and R. Eberhart (1995). *"Particle Swarm Optimization"*, Proc. IEE Int. Conf Neural Networks (ICNN'95), vol. IV, Perth, Australia, pp 1942 – 1948

R. Eberhart and J. Kennedy (1995), *"A new optimizer using particle swarm theory"* In Proceedings of the 6th International Symposium on Micro Machine and Human Science, Piscataway, NJ, IEEE Press, pp 39 -43

A. Mohais, R. Mendes, C. Ward, and C. Postoff, (2005), *"Neighborhood restructuring in particle swarm optimization"*, Proceedings of the 18th Australian Joint Conference on Artificial Intelligence, Berlin, pp 776 – 785.

M. Clerc and J. Kennedy (2002), *"The Particle Swarm-Explosion, Stability and Convergence in a Multidimensional Complex Space"*, IEEE Transactions on Evolutionary Computation, vol 6, num. 1, pp 58 - 73

Y. Shi, R. Eberhart (1999), *"Empirical study of particle swarm optimization"*, In: Proceedings of the 1999 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press, pp 1945 – 1950

Y. L.. Zheng, L.H. Ma, L.Y. Zhang, J.X. Qian, (2003) *"Empirical study of particle swarm optimizer with an increasing inertia weight"*, In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press, pp 221 - 226

R. Eberhart, Y. Shi (2001), *"Tracking and optimizing dynamic systems with particle swarms"*, In proceedings of the 2001 IEEE Congress on Evolutionary Computation, Pisctaway, NJ, IEEE Press, pp 94 - 100

R. Mendes, J. Kennedy, J. Neves, (2004), *"The Fully Informed Particle Swarm : Simpler, maybe better"*, IEEE Transactions on Evolutionary Computation, vol 8, no. 3, pp 204 - 210

A. Ratnaweera, S.K. Halgamuge, and H.C. Watson (2004), *"Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients"*, IEEE Transactions on Evolutionary Computation, Vol. 8, No. 3, pp 240 – 255

C.-C. Chen, (2009), "*Hierarchical Particle Swarm Optimization for Optimization Problems"*, Tamkang Journal of Science and Engineering, Vol. 12, No. 3, pp. 289 – 298

S. Janson, M. Middendorf, (2005), *"A Hierarchical Particle Swarm Optimizer and Its Adaptive Variant"*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 35, No 6, pp 1272 – 1282

M. Iqbal, Marco A. Montes, *"An Estimation of Distribution Particle Swarm Optimization Algorithm"*

Ferdi S., Achmad R., Iwan I., *Reduksi Suara Jantung dari Rekaman Suara Paru-Paru Menggunakan Filter Adaptif Dengan Algoritma Recursive Least Square*, Prosiding SENTIA 2009, Politeknik Negeri Malang, 2009

Sumardi, Syahid, *Simulasi Penekanan Derau dengan Metode Finite Impulse*

Respone (FIR) secara Adaptif Menggunakan Algoritma Least Mean Square (LMS), Fakultas Teknik Universitas Muhammadiyah Surakarta

Anita N., SariSukojo S., *Adaptive Noise Canceling* Menggunakan Algoritma *Least Mean Square (LMS)*, Jurnal Teknik Elektro Vol. 3, No. 1 Januari – Juni 2011

Uma R., *EHW Architecture for Design of FIR Filters for Adaptive Noise Cancellation*, IJCSNS International Journal of Computer Science and Network Security, Vol. 9, No. 1, Januari 2009, pp 41 - 49

Puja Pramudya, *Aplikasi Kriptografi Untuk Keamanan Pelaporan Pemungutan Suara Pada Pemilihan Umum Presiden Berbasiskan Layanan Pesan Pendek di Indonesia*

Supriyono, *Pengujian Sistem Enkripsi-Dekripsi Dengan Metoda RSA Untuk Pengamanan Dokumen*, JFN, Vol 2, No. 2, November 2008