BAB III

METODE PENELITIAN DAN PERANCANGAN SISTEM

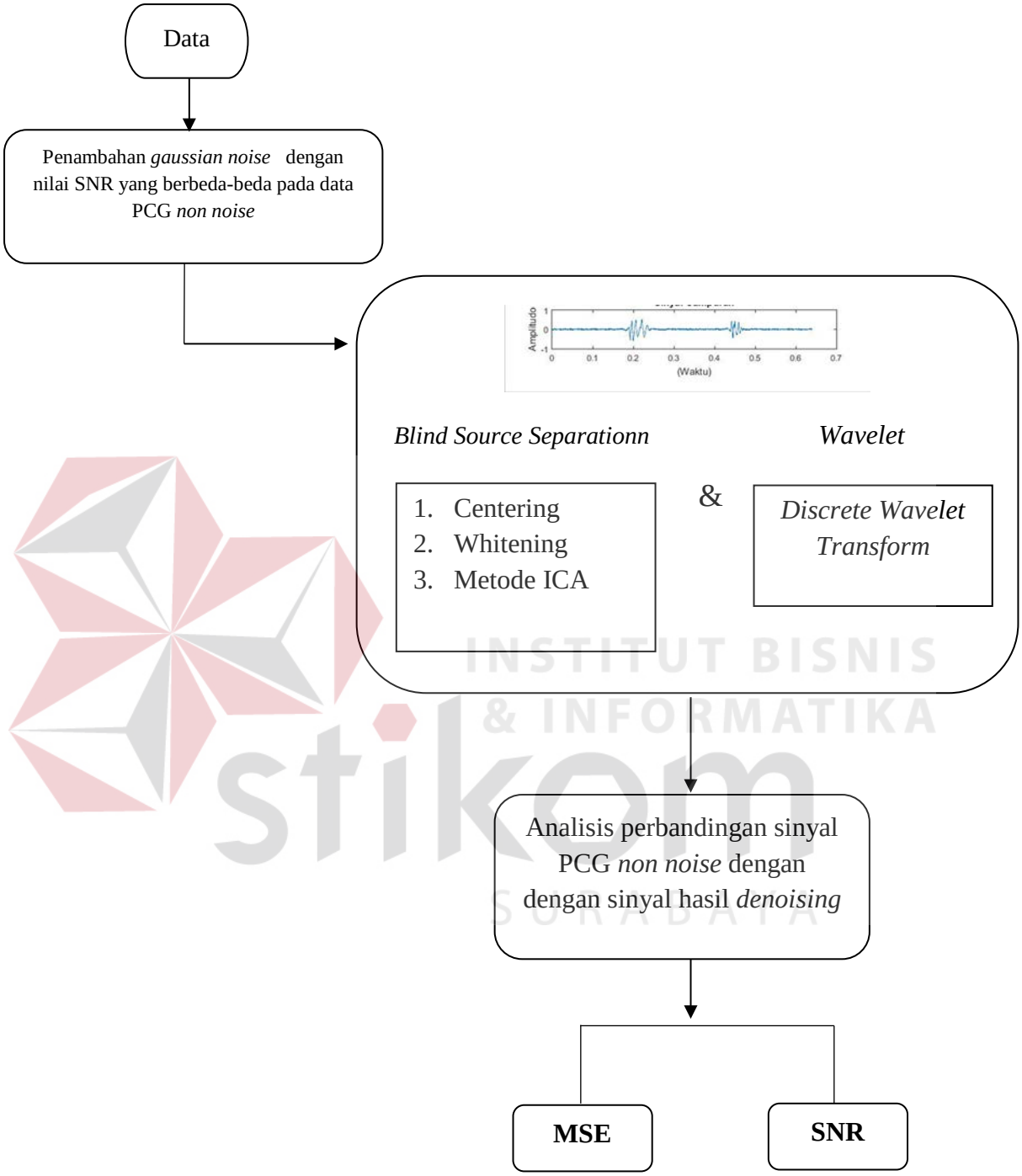
3.1 Metode Penelitian

Metode penelitian yang digunakan pada penelitian ini adalah *denoising* menggunakan *Blind Source Separation* dengan metode ICA. Data sinyal jantung yang digunakan dalam penelitian ini diambil dari tugas akhir Anggi Tiara, kemudian penambahan sinyal yang bersih dengan sebuah *noise Gaussian*, dan tahapan terakhir adalah tahapan *denoising* dengan metode ICA.

Tahapan untuk *preprocessing* terdiri atas 2 tahapan yaitu *Centering* dan *whitening*. *Centering* adalah sebuah proses untuk membuat rataan (*mean)* suatu data atau sinyal menjadi nol yakni dengan mengurangi sinyal tersebut dengan rataan dirinya sendiri. Sedangkan *whitening* adalah proses transformasi data atau sinyal kedalam bentuk data baru dimana nilai kovarian matriknya merupakan matrik identitas.

Setelah dilakukan tahapan *preprocessing* akan dilakukan proses *denoising* dengan metode ICA (*Independent Component Analysis*). Metode ICA adalah sebuah teknik untuk mengembalikan beberapa sinyal tercampur menjadi beberapa sinyal tercampur menjadi beberapa sinyal independen (Aichner, 2004).

Setelah dilakukan proses denoising tahap selanjutnya adalah menganalisa antara sinyal asli dengan sinyal hasil *denoising* dengan menggunakan parameter SNR dan MSE. Blok diagram rancangan penelitian dapat dilihat pada gambar 3.1



Gambar 3.1 Diagram Blok Rancangan Penelitian.

3.2 Data Sinyal PCG

Pada penelitian ini data yang digunakan sebanyak 3 sampel dan diambil dari penelitian Tugas Akhir dari Anggi Tiara

3.3 Penambahan *Gaussian Noise*

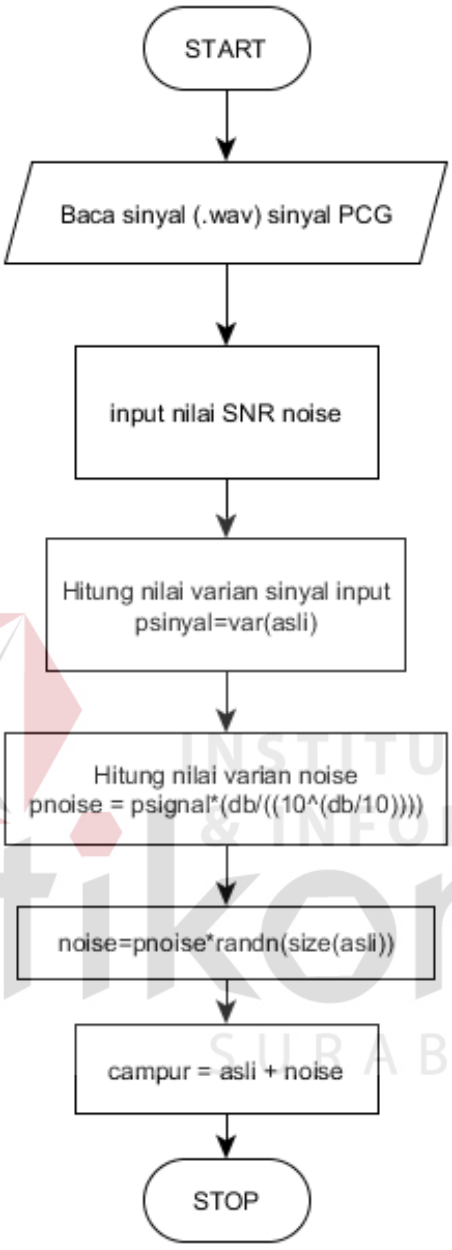
Setelah didapatkan sebuah data sinyal jantung, maka akan dilakukan proses penambahan *gaussian noise* dengan nilai nilai SNR yang berbeda – beda yaitu 5 dB, 10 dB, 15 dB, dan 20 dB. *Gaussian noise* adalah model noise yang mengikuti distribusi normal standar dengan rata-rata nol dan standar deviasi 1. Proses penambahan *gaussian noise* pada Matlab menggunakan fungsi **`randn`**, cuplikan program proses penambahan *gaussian noise* dapat dilihat pada gambar 3.3

```
18 -    noise=pnoise*randn(size(asli));
```

Gambar 3.2 Cuplikan Program Penambahan *Gaussian Noise*

`randn`, adalah fungsi di matlab untuk meng-generate distribusi normal / distribusi *Gaussian*

adapun *flowchart* penambahan *gaussian noise* proses penambahan *gaussian noise* dapat dilihat pada gambar 3.3



Gambar 3.3 *Flowchart* Program penambahan *gaussian noise* dan penambahan *noise* ke sinyal tanpa *noise*

Gambar 3.3 merupakan gambar *flowchart* dari program penambahan *gaussian noise* dan penambahan *noise* ke sinyal tanpa *noise* yang akan dijelaskan sebagai berikut :

1. Membaca file input berupa audio dari sinyal suara jantung (.wav)
2. Menginputkan SNR *noise*, di gunakan untuk penambahan *gaussian noise*
3. Menghitung varian dari sinyal tanpa *noise*
4. Mencari nilai varian dari *noise* dengan menggunakan rumus SNR yang telah di anti Log kan.
5. Penambahan *gaussian* dengan mengkalikan varian dari *noise* dengan fungsi `randn (size(asli))`. Size asli digunakan untuk menyamakan panjang *noise* dangan panjang sinyal tanpa *noise*.
6. Menjumlahkan sinyal tanpa *noise* dan sinyal *noise*.
7. Sinyal dan *noise* berhasil di campur.

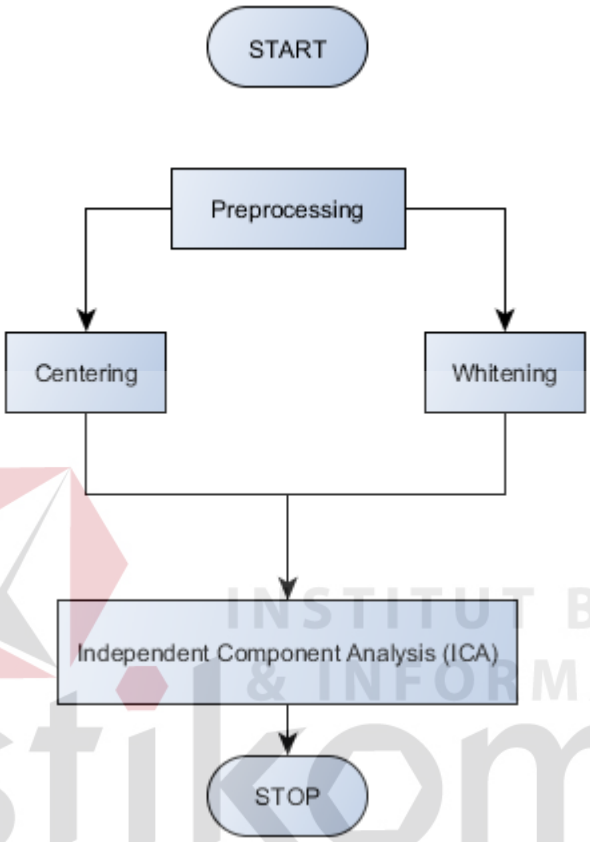
3.4 *Denoising* Sinyal PCG

Setelah dilakukan proses penambahan *noise* pada sebuah sinyal jantung tahapan selanjutnya yang akan dilakukan adalah proses denoising sinyal jantung. *Denoising* sinyal bertujuan untuk menghapus data yang tidak diperlukan, yang terekam pada saat proses perekaman suara sinyal jantung. Setelah sinyal suara jantung diambil maka sinyal harus di *denoising* terlebih dahulu. *Denoising* sinyal menggunakan 2 metode, diantaranya yaitu *Blind Source Separation* dan *Wavelet.*

3.4.1 *Blind Source Separation*

Blind Source Separation adalah pemisahan sinyal-sinyal yang tercampur tanpa mengetahui informasi yang dimiliki oleh sinyal-sinyal tercampur tersebut.

Tahapan untuk melakukan *denoising* dengan menggunakan *Blind Source Separation* dapat dilihat pada gambar 3.4.



Gambar 3.4 Blok Diagram *Blind Source Separation*

3.4.1.1 *Centering*

Centering adalah sebuah proses dimana bertujuan untuk membuat rataan suatu data menjadi nol dengan cara mengurangi sinyal tersebut dengan rataan dirinya sendiri. Dari proses pengurangan tersebut didapatkan sebuah data baru denga nilai rataa nol. cuplikan program dari proses centering dapat dilihat pada gambar 3.5

```
% Centering
xc = x - repmat(mean(x')',1,size(x,2));
```

Gambar 3.5 Cuplikan Program Centering

Fungsi **repmat** pada cuplikan program gambar 3.5 berfungsi untuk membuat sebuah matriks baru yang ukurannya sama dengan matriks sebelumnya.

3.4.1.2 *Whitening*

Whitening adalah sebuah proses mentransformasikan data ataupun sinyal ke bentuk data baru di mana nilai dari kovarian matriksnya adalah matriks identitas. Proses *whitening* berfungsi untuk me“mutih”kan variabel yang diamati, sehingga didapatkan sebuah vektor baru yang variansnya sama dengan satu serta komponen nyata dan imajinernya tidak terkorelasi dengan nilai varians yang sama. Pada tahapan ini data ditransformasikan sehingga didapatkan vektor data baru dengan karakteristik antara baris yang satu dan baris yang lainnya tidak saling berkorelasi, memiliki variasi yang sama, kovariansi matriks = matriks identitas [I]. Cuplikan program dari proses centering dapat dilihat pada gambar 3.6

```
[E, D] = eig(cov(xc,1));
xw = E * inv(sqrt (D)) * E' *xc;
```

Gambar 3.6 Cuplikan Program Whitening

Maksud dari cuplikan pada gambar 3.6 adalah setelah didapatkan matriks hasil proses centering akan dilakukan proses transformasi data ke bentuk data baru agar

didapatkan nilai matriks kovarianny berupa matriks identitas. Pada proses whitening ini digunakan fungsi pada MATLAB yaitu **cov**. Cov adalah sebuah fungsi pada MATLAB yang digunakan untuk menghitung nilai kovarian dari sebuah matriks. Pada gambar 3.6 terdapat fungsi cov(xc,1) maksudnya adalah Xc adalah matriks hasil proses centering dan 1 adalah angka yang digunakan agar didapatkan nilai variansnya sama dengan 1

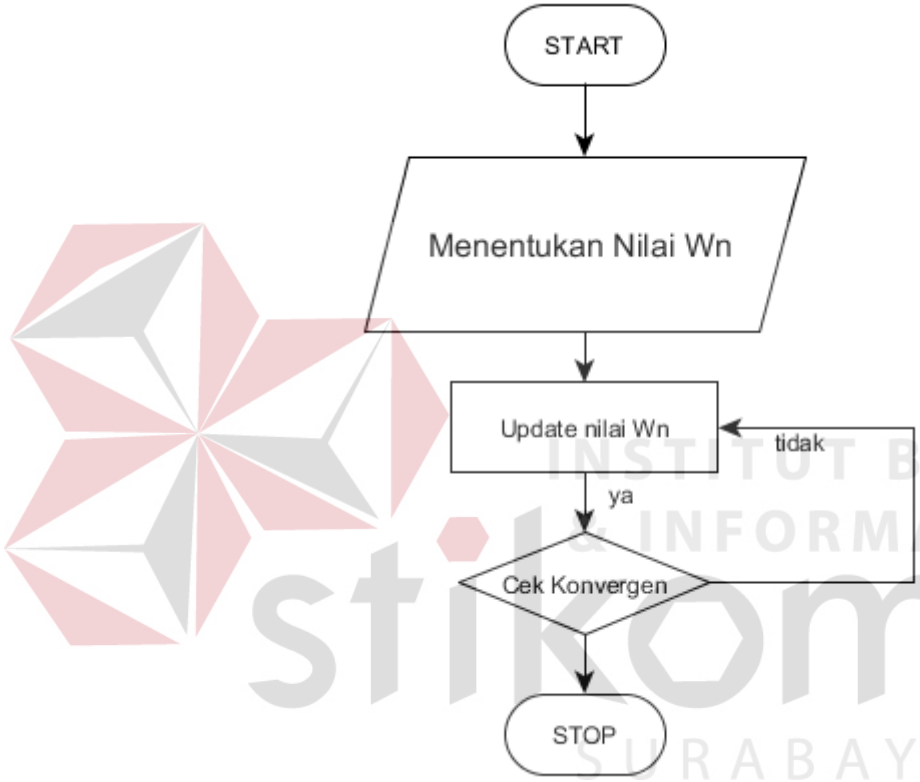
3.4.1.3 *Independent Component Analysis*

Pada ICA, setiap sinyal yang ditangkap/direkam merupakan hasil dari suatu fungsi linier. Dengan menganggap bahwa S_i bebas statistika, maka invers dari persamaan x=As dapat ditulis:

$$\mathbf{S=Wx} \qquad (3.1)$$

dimana W adalah matriks invers dari matriks pencampur A. Jika y adalah salah satu nilai komponen bebas dari mixed signal, atau y = WTn dan wi merupakan salah satu komponen baris dari matriks invers A, maka y adalah salah satu komponen bebas dari X. Permasalahan yang dihadapi sekarang adalah bagaimana cara untuk mendapatkan wi yang memiliki nilai yang sama dengan salah satu baris dari vektor A. Untuk melihat konsep dasar ICA, persamaan y = WTn = dituliskan kembali menjadi $z = A^Tx$, sehingga didapatkan y = WTn = $W^TAs = z^Ts$. Terlihat bahwa y merupakan kombinasi linear dari s dengan weightened factor. Karena jumlah dari dua komponen bebas atau lebih memiliki sifat gaussian yang lebih besar dari sinyal asli s, maka z^Ts lebih gaussian dari semua komponen s yang ada, dan nilai gaussian z^Ts akan semakin kecil dan

mendekati nilai s jika z^Ts mendekati atau sama dengan salah satu variabel bebas s. Berdasarkan hal ini, nilai dari vektor W dapat dicari dengan cara memaksimalkan gaussianity nilai WTn. *Flowchart* perhitungan dari vektor W (un*mixing matrix)* dapat dilihat pada gambar 3.7



Gambar 3.7 *Flowchart* Perhitungan Matriks Pemisahan Sinyal

Gambar 3.7 merupakan gambar *flowchart* dari perhitungan matriks pemisahan sinyal yang akan dijelaskan sebagai berikut :

1. Tentukan nilai Wn secara *random.* Nilai Wn sendiri adalah nilai awal vektor kompleks
2. Update atau perbaharui nilai Wn berdasarkan rumus berikut

$$w_+ = E\{X_w g(WT_n X_w)\} - E\{g'(WT_n X_w)\}\, w \qquad (3.2)$$

$$w_{n+1} = (w + w + T) - 12W + \quad (3.3)$$

Dimana nilai **g(u)** yang digunakan adalah sebagai berikut

$$g(u) = \tanh(a1u) \quad (3.4)$$

$$g'(u) = a1(1 - \tanh(a1u)\,2) \quad (3.5)$$

Variabel **g(u)** berfungsi untuk mencari sebuah nilai yang sama dengan nilai Wn = Wn +1

3. Cek konvergensi

$$1 - \min\big(diag(|WnWTn + 1|)\big) < eps \quad (3.6)$$

Jika belum konvergen, kembali ke langkah 2 dimana nilai dari Wn adalah Wn = Wn + 1

3.4.2 *Wavelet*

3.4.2.1 Denoising

Data sinyal PCG yang telah tercampur *noise* akan dicari nilai *threshold* dengan metode *threshold rules* yang ada. Nilai *threshold* digunakan untuk perbandingan pada setiap koefisien *wavelet*. *Denoising* digunakan untuk menghapus data sinyal yang tidak diperlukan dengan membandingkan nilai *threshold* dan setiap koefisen *wavelet*, yang telah ditambaakan *guassian noise*. Metode *Denoising* yang digunakan adalah *soft tresholding*, dimana metode ini akan membuat nilai yang berada antara *threshold* –T< X < T menjadi perlahan menuju 0, sedangkan nilai yang lebih dari T telah diubah untuk mendekati axis X. *Denoising* pada penelitian ini dilakukan secara *adaptive* karena *threshold* yang didapatkan dari karakteristik dari sinyal input.

Pada penelitian yang dilakukan mencari nilai *threshold* dengan karakteristik sinyal input dengan penerapan *thresholding rules* terdapat 2 metode yaitu *global thresholding* dan *level dependent thresholding*. Pencarian nilai *threshold* dengan *global thresholding* menggunakan karakteristik dari panjang data dari sinyal input. Sedangkan *level dependent thresholding* mencari nilai *threshold* berdasarkan level resolusi / level dekomposisi. Pada penelitian ini level resolusi yang digunakan adalah 10 level dikarenakan sinyal yang diolah memiliki frekuensi sampling sebesar 8KHz. Pada penelitian ini penggunaan metode *level dependent thresholding* digunakan *function* pada matlab. Cuplikan program proses pemanggilan *function* dapat dilihat pada Gambar 3.8.

```
xd = wden(campur,'sqtwolog','s','sln',10,'db5');
```

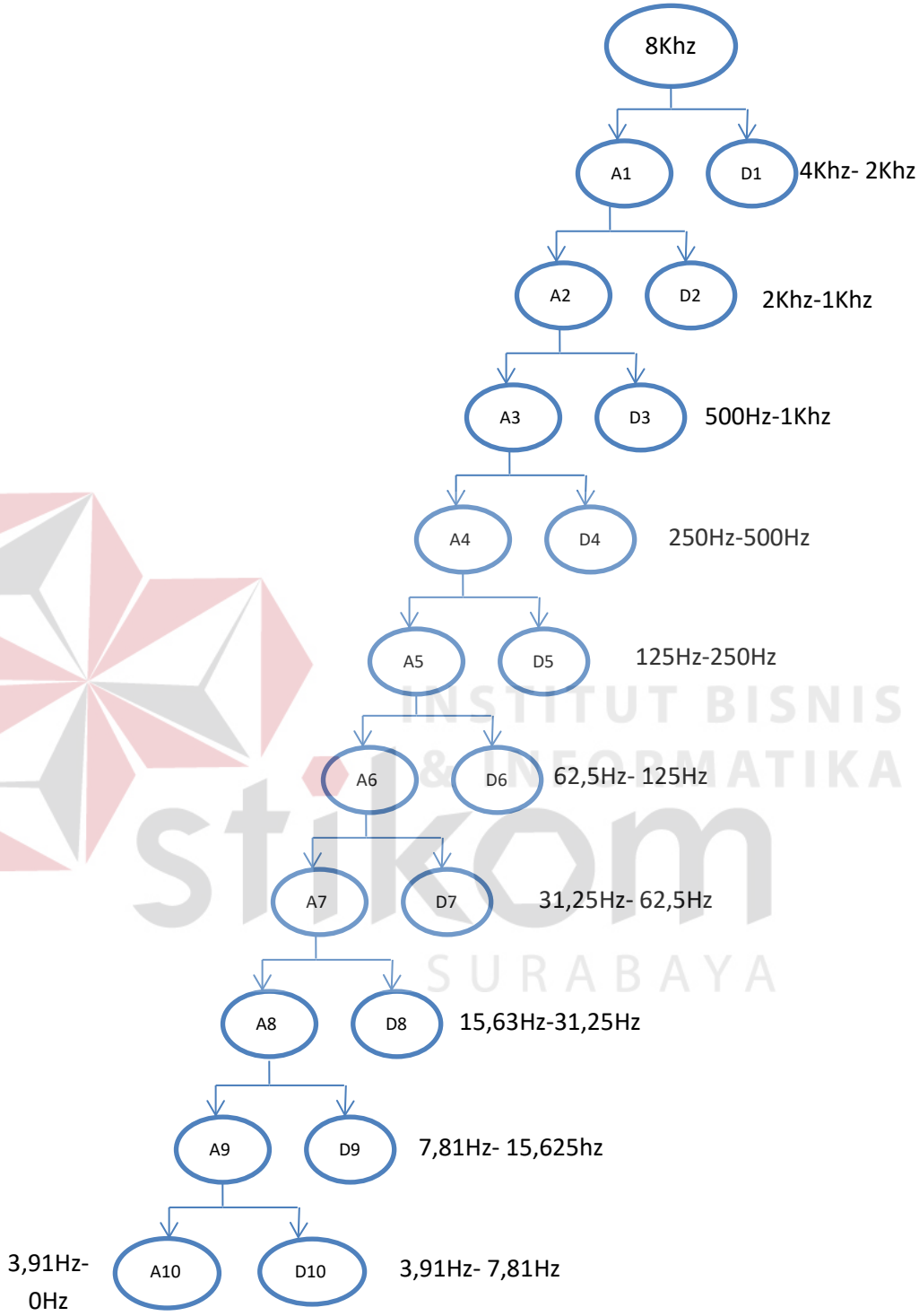
Gambar 3.8 Pemanggilan *Function Level Dependent Thresholding*

Pada matlab terdapat *function* yang digunakan untuk memanggil *level dependent thresholding* yaitu `wden`. Pada function ini memiliki parameter input sinyalnoise untuk sinyal masukan, thrules untuk metode *level dependent thresholding*, level untuk tingkatan level, dan ‘w’ untuk Mother Wavelet,

3.4.2.2 *Discrete Wavelet Transform* (DWT)

Pada penelitian ini digunakan analisis Transformasi *Wavelet* Diskrit. Transformasi *Wavelet* diskrit digunakan untuk mendekomposisikan sinyal masukan PCG ke dalam bentuk gelombang seusai dengan *Mother Wavelet* yang digunakan, dekomposisi dilakukan dengan memisahkan sinyal masukan ke dalam frekuensi

rendah dan frekuensi tinggi, hasil dari dekomposisi adalah komponen *approximation* yang merupakan *scaling function* (*lowpass filter*) dan komponen *detail* yang merupakan *Wavelet function*. (Sundararajan, 2015).



Gambar 3.9 Dekomposisi 10 Tingkat Dengan Frekuensi Cuplik 8KHz.

Level dekomposisi ditetapkan berdasarkan frekuensi cuplik yang digunakan. (Venkatta, 2015). Penelitian ini dipengaruhi beberapa parameter yaitu sinyal PCG dari setiap subyek, frekuensi cuplik, *Mother Wavelet*, dan level dekomposisi. Sinyal PCG akan didekomposisikan menjadi A yang merupakan approksimasi dan D yang merupakan detail, serta akan didekomposisikan sesuai dengan frekuensi cuplik 8Khz akan didekomposisikan sebanyak 10 tingkat

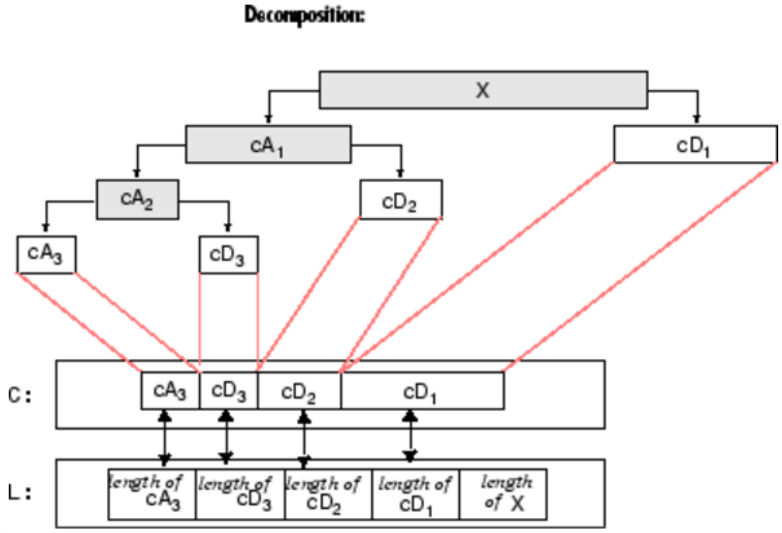
Analisis transformasi *Wavelet* diskrit dilakukan dengan mendekomposisi sinyal PCG menggunakan Matlab, untuk mendekomposisi sinyal satu dimensi maka digunakan fungsi `wavedec`, cuplikan program proses dekomposisi dapat dilihat pada Gambar 3.10

```
data=asli;
level=10;
n=level;
w='db5'

%decomposition
[C,L]=wavedec(data,n,w);
ca10=appcoef(C,L,w,n);
[cd1,cd2,cd3,cd4,cd5,cd6,cd7,cd8,cd9,cd10]=detcoef(C,L,[1,2,3,4,5,6,7,8,9,10]);
```

Gambar 3.10 Cuplikan Program Proses Dekomposisi.

Fungsi `wavedec('x',N,'Wname')` pada matlab memiliki parameter input x untuk sinyal masukan, N untuk tingkat level, dan Wname untuk *Mother Wavelet*, sedangkan parameter outputnya adalah hasil dekomposisi dan panjang data dari setiap komponen dapat dilihat pada Gambar 3.11

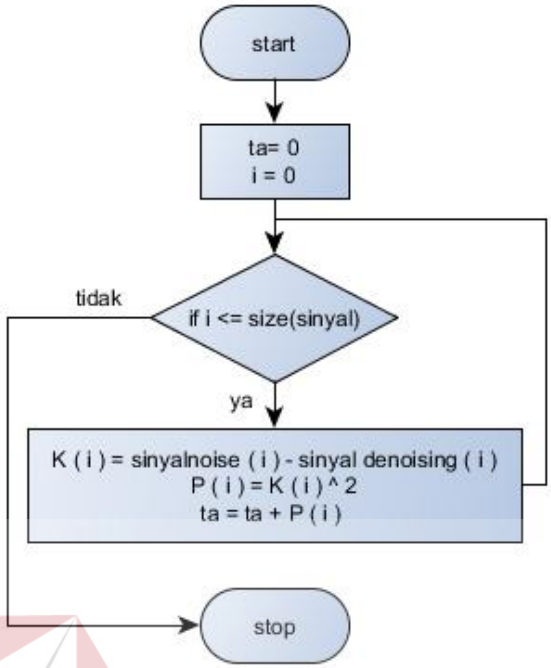


Gambar 3.11 Dekomposisi *Wavelet* Diskrit 1D. (Matlab)

3.5 Perhitungan MSE dan SNR

3.5.1 Perhitungan MSE

Proses ini bertujuan untuk mendapatkan nilai MSE yang dianalisis. *Flowchart* perhitungan nilai MSE dapat dilihat pada gambar 3.12



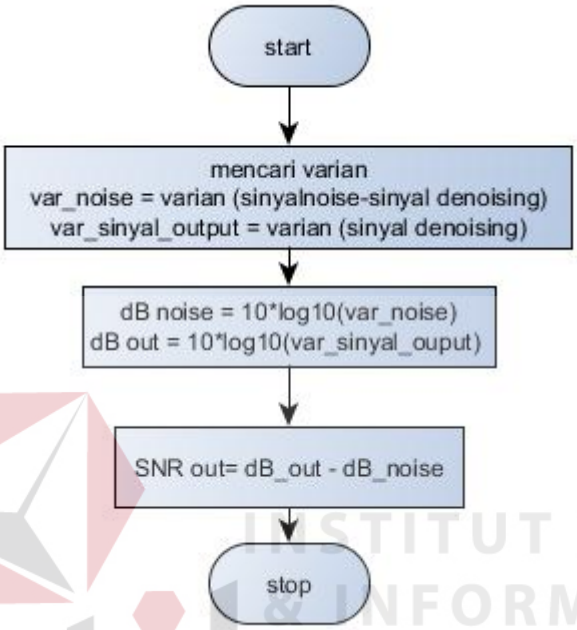
Gambar 3.12 *Flowchart* program untuk mencari nilai MSE

Gambar 3.12 merupakan gambar *flowchart* dari program Perhitungan nilai *Mean Square Error* (MSE) yang akan dijelaskan sebagai berikut:

1. Inisialisasi ta = 0 dan i = 0;
2. Melakukan perulangan selama i kurang dari panjang data sinyal.
3. Mengkurangi array isi array sinyalnoise (indeks ke-i) dengan isi array sinyal denoising (indeks ke-i).
4. Mengkuadratkan hasil pengurangan isi dari tiap array.
5. Menjumlahkan semua hasil kuadrat.
6. Mendapatkan nilai hasil MSE.

3.5.2 Perhitungan SNR

Proses ini bertujuan untuk mendapatkan nilai SNR, guna mengetahui kualitas hasil proses. *Flowchart* perhitungan nilai MSE dapat dilihat pada gambar 3.13



Gambar 3.13 *Flowchart* program untuk mencari nilai SNR

Gambar 3.13 merupakan gambar *flowchart* dari program Perhitungan nilai *Signal to Noise ratio* (SNR) yang akan dijelaskan sebagai berikut:

1. Mencari varian dari noise dan varian dari sinyal yang telah di denoising
2. Mengkalikan rumus SNR dengan varian dari noise dan varian sinyal yang telah di denoising.
3. Mengkurangi hasil SNR sinyal output dengan SNR sinyal noise.
4. Nilai SNR telah di dapatkan.