BAB 3
METODE PENELITIAN

3.1 ALAT YANG DIGUNAKAN

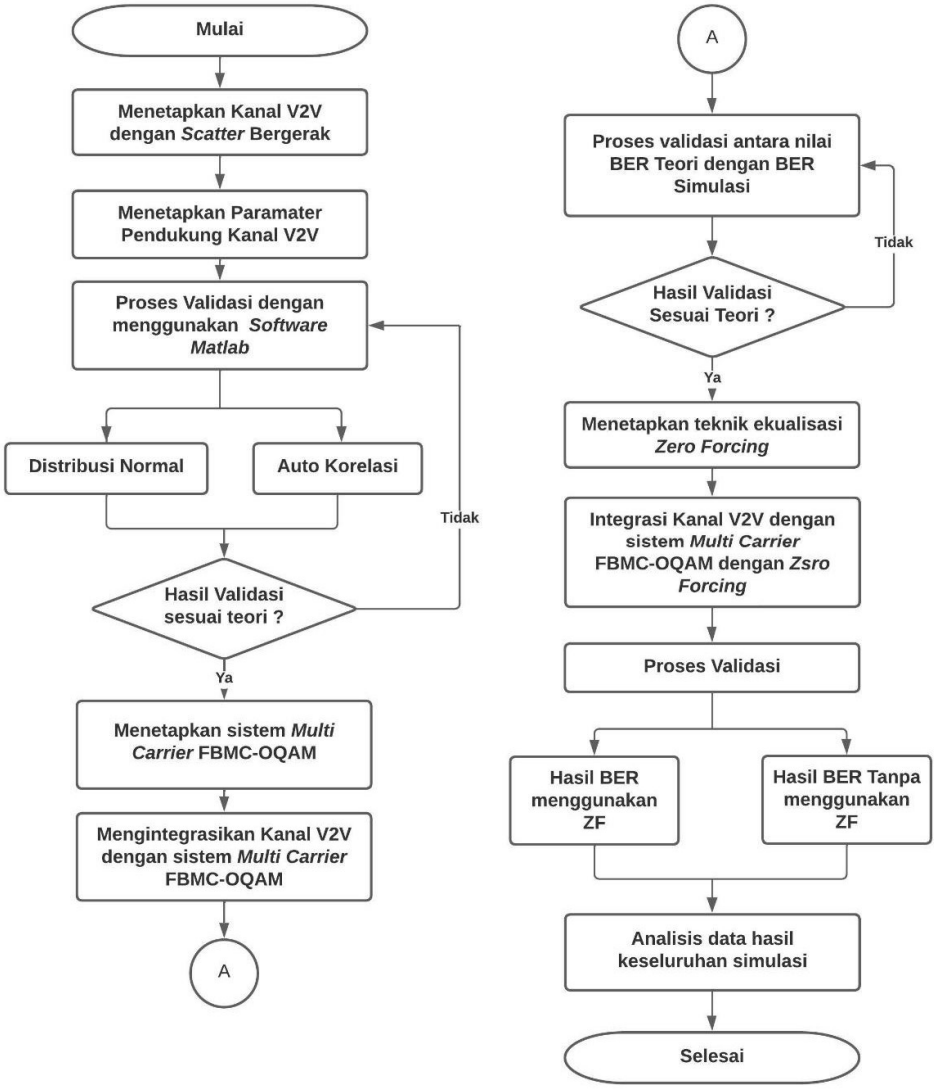
Penelitian ini akan digunakan pemodelan kanal V2V menggunakan *scatterers* bergerak. Kecepatan kendaraan yang digunakan dengan kecepatan rendah, sedang dan cepat. Jumlah *scatterer* yang digunakan sebanyak delapan. Sistem *mulitcarrier* yang digunakan adalam FBMC. Untuk memitigasi efek *Doppler* yang terjadi dengan menambahkan metode ekualisasi *Zero Forcing* dan memvalidasi hasil integrasinya dalam parameter *Bit Error Rate*. Untuk mengimplementasikan simulasi pada penelitian ini digunakan *software* Matlab R2018a.

3.2 ALUR PENELITIAN

Rancangan proses pengerjaan pada penelitian ini, mengenai simulasi program kanal V2V menggunakan sistem *multicarrier* FBMC-OQAM dengan modulasi 16QAM, dimana pada *multicarrier* FBMC menggunakan *filter* Ideal. Mengacu pada diagram alir yang ditunjukkan pada Gambar 3.1.

Gambar 3.1 menunjukkan proses perancangan alur peneltian yang dilaksanakan. Pada bagian pertama yaitu menetapkan rancangan jenis kanal V2V dengan *scatterer* bergerak sebagai kanal yang digunakan. Selanjutnya menetapkan parameter-parameter pendukung kanal V2V yaitu berupa kendaraan disisi *transmitter* dan *receiver* dengan kecepatan rendah, kecepatan sedang, kecepatan tinggi, dan jumlah *scatterers* yang bergerak yang berada disekitar area kanal. Proses validasi dengan *software* matlab menggunakan distribusi normal dan autokorelasi. Kemudian setelah proses validasi sesuai maka selanjutnya menetapkan *multicarrier* yang digunakan yaitu FBMC-OQAM dengan modulasi 16QAM yang didalamnya terdapat *filter* ideal di sisi blok pengirim dan blok penerima. Di sisi penerima nanti terdapat proses ekualisasi linier yaitu *Zero Forcing*. Kemudian dari seluruh blok tersebut yaitu kanal V2V, *Multicarrier* FMBC, dan juga *Zero Forcing* diintegrasikan yang kemudian sistem tersebut diuji pengaruhnya berdasarkan

parameter BER hasil simulasi dan teori. Jika hasil keluaran tidak sesuai maka dilakukan perbaikan program dan pengujian program. Sebaliknya, jika hasil keluaran sesuai dengan parameter BER maka hasil keluaran akan dilakukan analisis. Terlihat pada Gambar 3.1 merupakan *flowchart* rancangan simulasi.



Gambar 3.1 *Flowchart* Rancangan Simulasi

3.3 PARAMETER SIMULASI

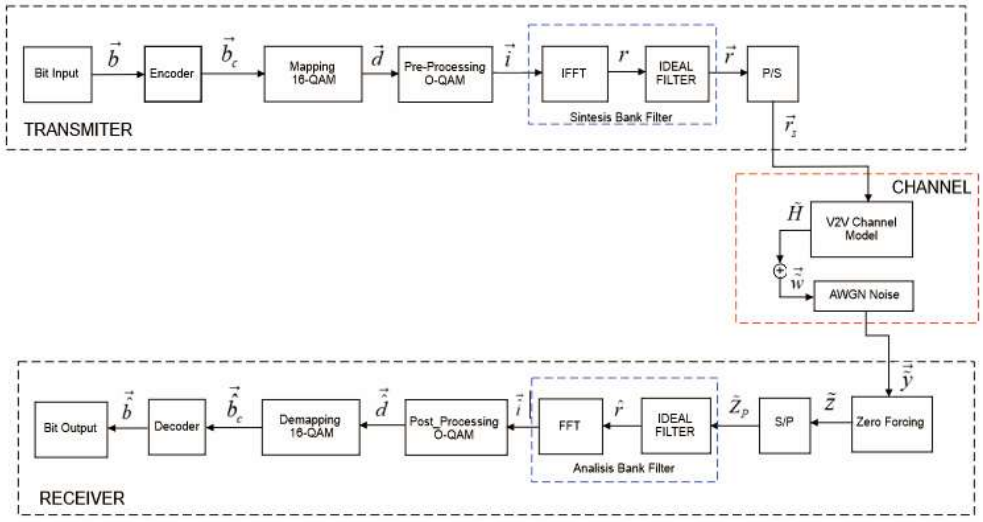
Paramater yang digunakan dalam simulasi program, tertera pada Tabel 3.1.

Tabel 3. 1 Parameter Simulasi

Parameter	Simbol	Nilai
Jenis Modulasi	*mdls*	16-QAM
Jumlah Level Modulasi	ml	4
Bit *Input*	b	2048
Filter	r	Filter Ideal
Kanal	h	V2V
Frekuensi Pembawa	f_c	$5{,}8x10^9$ Hz
Frekuensi Sampling	f_s	22050 Hz
Kecepatan Kendaraan	$V_t = V_r$	10 m/s, 50 m/s dan 100 m/s
Kecepatan *Scatterer*	V_s	5 m/s
Jumlah *Scatterer*	n	8
Sudut antara garis horizontal dengan arah pergerakan kendaraan Tx	αvt	5°
Sudut antara garis horizontal dengan arah pergerakan kendaraan Rx	αvr	5°
Fase	θ	5°

3.4 PEMODELAN SISTEM

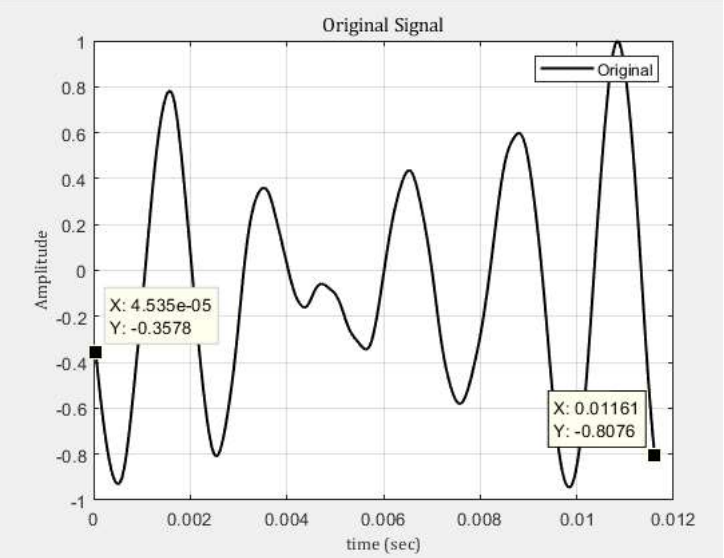
Sebagai tahap awal penelitian ini adalah merancang pemodelan sistem sebagaimana terlihat pada Gambar 3.2 berikut.



Gambar 3.2 Pemodelan Sistem

3.4.1 Data *Input*

Proses pertama data masukan pada simulasi ini berupa sinyal audio dalam bentuk *.wav dengan jumlah *bit input* ($\vec{b}$) sebanyak 2048-bit atau sekitar 0,012 detik. Sinyal tersebut menunjukkan adanya fluktuasi nilai yang acak sehingga sesuai untuk simulasi yang akan dijalankan.



Gambar 3.3 Data *Input*

Untuk dapat mengubah titik tersebut menjadi *biner*, amplitudonya perlu ditambahkan dengan 1 terlebih dahulu agar menjadi positif. Pada Gambar 3.3 sebuah titik sinyal audio dengan waktu (t_{at}) 4,535e-05 detik memiliki amplitudo sebesar -0,3578 volt. Jika *amplitude* pada waktu tersebut dilambangkan dengan simbol A_t, maka mengubah A_t tersebut menjadi bilangan biner adalah sebagai berikut.

$$A_t = (\text{A}+1) \text{ x } 2^{n-1}$$

$$A_t = (-0{,}3578+1) \text{ x } 2^{8-1}$$

$$A_t = 0{,}6422 \text{ x } 2^{7}$$

$$A_t = 0{,}6422 \text{ x } 128$$

$A_t = 82{,}2016$

$A_t = 82_{10}$

Kemudian hasil bilangan desimal tersebut diubah menjadi bilangan biner dengan cara:

82/2 = 41 sisa 0

41/2 = 20 sisa 1

20/2 = 10 sisa 0

10/2 = 5 sisa 0

5/2 = 2 sisa 1

2/2 = 1 sisa 0

1/2 = 0 sisa 1

Maka, hasil binernya menjadi $0101\ \ 0010_{(2)}$. Hasil biner inilah yang akan dikirim dan akan diproses pada blok selanjutnya.

Gambar 3.4 Data *Input Biner*

3.4.2 *Encoder*

Proses selanjutnya data *bit input* $(\vec{b})$ selanjutnya akan dilakukan proses pengkodean $(\vec{b_c})$ dari serial ke paralel menggunakan modulasi dasar 16 QAM. Setiap simbol yang dikirimkan oleh modulasi 16 QAM terdiri dari 4 bit. Proses serial to paralel ini bertujuan untuk membagi data biner yang awalnya berbentuk seri menjadi bentuk paralel yang terdiri dari simbol-simbol yang dihasilkan. Seperti data biner

0101 0010 yang didapat dari hasil pengubahan data masukan, kemudian bit tersebut dibagi menjadi bentuk paralel 4 *bit* hingga hasilnya menjadi:

0101 (Simbol pertama)

0010 (Simbol kedua)

Untuk dapat mengetahui urutan indeks dari simbol tersebut, dapat diperolah dengan cara mengalikan frekuensi cuplik (fs) dengan waktu (t_{at}) sebagai berikut:

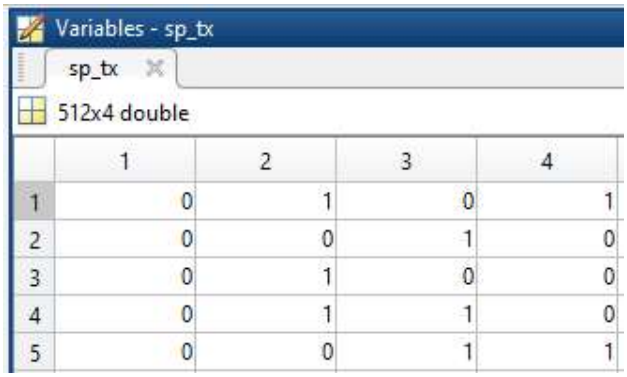
n = fs x t_{at}

n = 22050 x 0,00004535

n = 0,9999675

n = 1

Jadi, untuk simbol 0101 dan 0010 terletak pada urutan indeks ke-1 dan ke-2, seperti yang ditunjukkan oleh Gambar 3.5. Simbol-simbol yang telah didapat, selanjutnya akan diproses pada blok pemetaan 16 QAM.



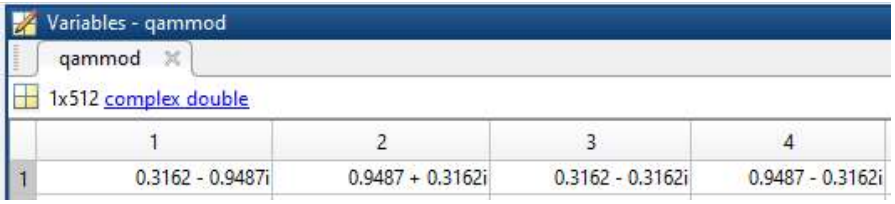
	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	1

Gambar 3.5 Seri ke Paralel

3.4.3 *Symbol Mapping* 16 QAM

Pemetaan simbol 16 QAM adalah proses merubah bentuk dari biner ($\vec{d}$) menjadi bilangan kompleks dengan rumus $S_k = I_k + jQ_k$, dimana *k* adalah variabel simbol, *I* merupakan *inphase* atau bilangan *riil* dan *Q* merupakan *quadrature* atau bilangan imajiner. Modulasi dasar yang dipergunakan dalam simulasi ini adalah 16

QAM dimana pemetaannya terdiri dari empat bit. Gambar 3.6 merupakan bentuk simbol *mapping* modulasi 16 QAM dalam bentuk bilangan komplek.



	1	2	3	4
1	0.3162 - 0.9487i	0.9487 + 0.3162i	0.3162 - 0.3162i	0.9487 - 0.3162i

Gambar 3.6 *Symbol Mapping* 16 QAM

3.4.4 *Pre-Processing* OQAM

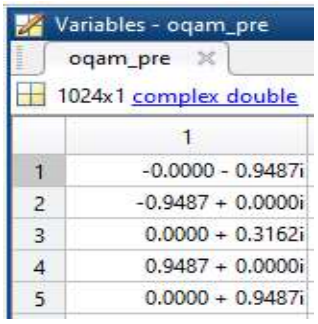
Simbol yang dikirimkan pada proses pra pengolahan O-QAM (i) dipecah menjadi 2 yaitu simbol ganjil dan genap. Proses pengolahan simbol terdiri dari 2 proses yaitu:

1. Proses pengubahan bilangan kompleks menjadi *riil*.
2. Proses perkalian dengan θ_{kn}

Pada Proses pertama bertujuan untuk membuat simbol baru. Simbol ganjil dan genap masing-masing akan dipisah menjadi bilangan *riil* dan kompleks. Masing-masing bilangan *riil* dan kompleks akan mengalami peningkatan jumlah (*upsampling* sebesar 2 kali) dan bergeser setengah fasa sebesar 90^o (1 fasa =180^o). Yang membedakan dalam proses pra pengolahan terletak pada Langkah pergeseran fasa. Simbol genap terjadi pergeseran fasa pada bilangan imajiner. Sedangkan simbol ganjil terjadi pergeseran fasa pada bilangan *riil*. Kemudian Langkah selanjutnya simbol yang dihasilkan pada proses pertama dikalikan dengan θ_{kn} untuk mendapatkan nilai aslinya dengan persamaan 3.4[11]:

$$\theta_{kn} = j^{(k+n)} \tag{3.4}$$

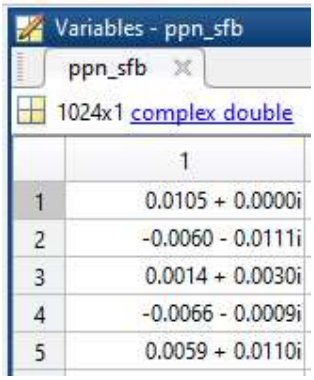
Dengan j adalah menunjukkan bilangan imajiner, k adalah urutan kanal, dan n aadalah jumlah kanal. Pada simbol genap dan ganjil terdapat pergeseran setengah fasa. Simbol genap memiliki pergeseran setengah fasa pada bilangan imajiner, sedangkan simbol ganjil terjadi pergeseran setengah fasa pada bilangan *riil*.



Gambar 3.7 Hasil Pra Pengolahan 16 QAM

3.4.5 *Syntesis Bank Filter*

Proses FBMC di sisi pengirim dinamakan dengan *syntesis bank filter*. Pada *syntesis bank filter* terdiri dari 2 proses yaitu kebalikan dari metode transformasi *fouirer* atau disebut dengan *Invers Fast Fourier Transform* (IFFT) dan proses *filter*. Hasil keluaran dari pra pengolahan OQAM dikenakan proses IFFT. Kemudian pada proses *filter*, keluaran dari IFFT (r) masing-masing dikalikan dengan *filter*. Untuk jenis *filter* yang digunakan pada simulasi ini diasumsikan sebagai *filter* ideal, maka hasil keluaran IFFT akan langsung diteruskan tanpa adanya penyaringan data terlebih dahulu. Sehingga hasil keluaran akhir ($\tilde{r}$) dari proses *syntesis bank filter* adalah seperti yang diperlihatkan pada Gambar 3.8 dibawah ini.



Gambar 3.8 Hasil *Syntesis Bank Filter*

3.4.6 Paralel to Serial (P/S)

Pada proses ini, data yang ditransmisikan dari bentuk paralel menjadi bentuk serial. Data yang telah melewati *filter* ($\tilde{r}$) yang berbentuk paralel diubah menjadi bentuk seri ($\tilde{r}_s$). Hal ini dilakukan agar keluaran dari proses *syntesis bank*

filter dapat dikirimkan menjadi 1 baris. Pada modulasi 16 QAM data keluaran dari *syntesis bank filter* berbentuk parelel 1024x1 bit. Kemudian dalam proses ini diubah menjadi bentuk serial dengan nilai keluaran ($\tilde{r}_s$) seperti pada Gambar 3.9 data tersebut nantinya akan transmisikan ke kanal V2V.

source

1x1024 complex double

	1	2	3	4	5
1	0.0105 + 0.0000i	-0.0060 - 0.0111i	0.0014 + 0.0030i	-0.0066 - 0.0009i	0.0059 + 0.0110i

Gambar 3.9 Hasil keluaran Paralel to Serial

3.4.7 Kanal V2V

Pemodelan kanal pada simulasi ini menggunakan kanal V2V ($\tilde{H}$) dengan *scatterers* bergerak. Pada kanal V2V mempunyai dua sisi yaitu pengirim (Tx) dan penerima (Rx) yang bergerak dengan kecepatan konstan dengan dikeliling *scatterers* bergerak. Pengirim dan Penerima beserta *scatterers* saling bergerak dengan kecepatan acak. Pada kanal V2V yang memiki arah propagasi dibagi menjadi 2 dimana sinyal yang dikirim dari penerima ke pengirim tanpa halangan atau LOS dan sinyal yang dikirim oleh penerima ke pengirim dipantulkan terlebih dahulu karena ada *scatterers* bergerak yang kemudian sinyal tersebut diteruskan ke penerima. Pada simulasi ini menggunakan kecepatan *vehicle* dengan kecepatan rendah 10 m/s, kecepatan sedang 50 m/s dan kecepatan tinggi 100 m/s. Jumlah *scatterers* yang digunakan sebanyak 8. Adapun untuk validasi kanal ini menggunakan proses distribusi *gaussian* atau distribusi normal.

xt

1x1024 complex double

	1	2	3	4	5
1	7.4065 + 0.0000i	7.5385 + 0.3154i	6.7634 - 0.1213i	6.1611 - 0.4592i	6.0189 - 0.9426i

Gambar 3.10 Hasil Perkalian Kanal dan *Multicarier*

3.4.3 Blok *Receiver*

3.4.3.1 *Zero Forcing*

Proses pengolahan sinyal pada deteksi simbol ($\tilde{y}$) bertujuan untuk mendapatkan nilai sinyal aslinya secara digital. Pada simulasi ini menggunakan algoritma *Zero Forcing* (ZF) karena algoritma ini kompleksitasnya rendah dengan menginverskan kanal H dan dikalikan dengan sinyal yang dikirim.

rx3

1x1024 complex double

	1	2	3	4	5
1	0.0227 - 0.0182i	-0.0233 - 0.0084i	-0.0003 + 0.0120i	-0.0134 + 0.0022i	0.0059 + 0.0205i

Gambar 3.11 Keluaran *Zero Forcing*

3.4.3.2 Serial to Paralel (S/P)

Proses pengubahan data seri menjadi paralel ($\tilde{z}$) berfungsi untuk mengubah bentuk keluaran dari antena penerima ke bentuk paralel ($\tilde{z}_p$) kemudian dilanjutkan dengan proses berikutnya yaitu proses analisis *bank filter*.

sp_rx2

1024x1 complex double

	1
1	0.0227 - 0.0182i
2	-0.0233 - 0.0084i
3	-0.0003 + 0.0120i
4	-0.0134 + 0.0022i
5	0.0059 + 0.0205i

Gambar 3.12 Nilai keluaran dari serial to paralel

3.4.3.3 Analisis *Bank Filter*

Pada analisis *bank filter* terdiri dari dua proses yaitu proses transformasi *fourier* dan proses *filter* atau *bank filter*. Pada simbol keluaran dari data seri ke paralel ($\tilde{z}_p$) kemudian dikalikan dengan *filter* dan menggunakan *filter ideal*. Sehingga simbol langsung di transmisikan tanpa adanya data yang hilang. Hasil

keluaran dari *filter* ($\hat{r}$) dikalikan dengan proses FFT. Hasil simbol keluaran dari FFT ($\vec{\hat{i}}$) dikirimkan untuk proses selanjutnya.

ppn_afb_out2

512x2 complex double

	1	2
1	0.0839 - 0.6475i	0.1354 - 0.0420i
2	-0.9988 + 0.1171i	-0.0650 + 0.0885i
3	0.2309 + 0.1993i	0.0514 - 0.3309i
4	1.1047 - 0.2663i	-0.3726 - 0.5126i
5	0.2659 + 0.8190i	-0.1193 + 0.8888i

Gambar 3.13 Hasil *Output* Analisis *Bank Filter*

3.4.3.4 *Post Processing* OQAM

Post Processing ($\vec{\hat{d}}$) merupakan kebalikan dari *Pra Processing* OQAM. Pada proses pasca pengolahan OQAM terdiri dari dua proses yaitu:

1. Melakukan perkalian dengan kebalikan θ_{kn}
2. Mengubah bilangan *riil* menjadi bilangan kompleks

Dalam proses pasca pemrosesan OQAM adalah Langkah pertama merupakan perkalian dengan *conjugate* dari θ_{kn}. Operasi kedua bertujuan untuk menurunkan laju *sample* dengan faktor 2. Pada proses kedua mengubah bilangan *riil* menjadi kompleks.

oqam_post2

512x1 complex double

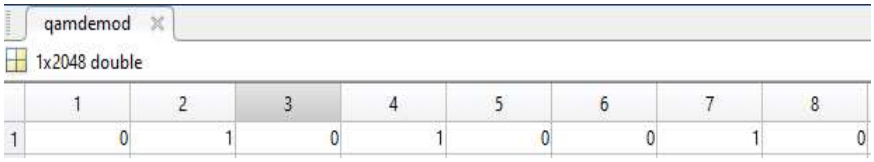
	1
1	-0.0420 - 0.6475i
2	0.9988 + 0.0650i
3	0.3309 - 0.1993i
4	1.1047 - 0.3726i
5	0.8888 + 0.8190i

Gambar 3.14 Hasil OQAM *Post Processing*

3.4.3.5 *Symbol Demapping*

Pada sisi penerima, proses simbol *demapping* merupakan proses kebalikan (*invers*) dari *mapping*. Dalam pengolahan ini, simbol akan dilakukan pemetaan

ulang untuk membentuk ukuran bit data yang sesuai dengan level modulasi. Proses *demapping* pada variasi modulasi QAM difungsikan untuk memilih simbol mana yang harusnya dikirim dari antena *transmitter*. *Demapping* dibutuhkan disebabkan oleh efek kanal serta *noise* yang menyebabkan simbol QAM dibagian *receiver* tidak menjadi asli dibandingkan *output mapping* QAM di *transmitter*. Adapun hasil keluaran dari proses ($\vec{\hat{b}}_c$) tersebut berupa komponen *riil* dan imajiner yang akan dikembalikan kembali sebagai bit data.



Gambar 3.15 Hasil *Output* keluaran QAM *Demapping*

3.4.3.6 Data *Output*

Data keluaran ($\tilde{\hat{b}}$) berupa *bit-bit* biner yang sudah dikembalikan ke bentuk semula. Hasil sinyal keluaran tersebut dianalisis dengan membandingkan nilai BER simulasi dengan BER teori. Proses selanjutnya data *output* berupa sinyal yang diterima diubah kembali ke dalam *range* -1 *Volt* hingga 1 *Volt*. Pengubahan ini merupakan kebalikan dari proses sebelumnya yang di dapat dilihat pada persamaan berikut[10]:

$$V_r = decimal(B) \tag{3.6}$$

$$V_1^{'} = V_r / 2^{nbit-1} \tag{3.7}$$

$$V_0^{'} = V_1^{'} - 1 \tag{3.8}$$

Dengan B merupakan deret biner yang diubah ke dalam bilangan desimal menggunakan operasi desimal dan disimpan dalam variabel V_r. Kemudian hasil tersebut dibagi dengan level kuantisasinya (n*bit*) dan disimpan dalam variabel $V_1^{'}$.

Hasil akhir adalah variabel $V_o^{'}$ yang merupakan pengurangan 1 dari variabel $V_1^{'}$. Sehingga didapat sinyal dalam rentang -1 hingga 1 *volt*.


data_output

1x2048 double

	1	2	3	4	5	6	7	8
1	0	1	0	1	0	0	1	0

Gambar 3.16 Data *Output*