TUGAS AKHIR – EE 184801

ANALISIS KINERJA SISTEM POINT-TO-POINT LARGE-SCALE ARRAY MIMO BERBASIS HYBRID BEAMFORMING PADA KANAL RAYLEIGH DAN UR-LOS

Putu Agus Kartika Adhi Pratama
NRP 07111640000147

Dosen Pembimbing
Dr. Ir. Puji Handayani, M.T.
Prof. Ir. Gamantyo Hendrantoro, M.Eng.,Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – EE 184801

ANALISIS KINERJA SISTEM POINT-TO-POINT LARGE-SCALE ARRAY MIMO BERBASIS HYBRID BEAMFORMING PADA KANAL RAYLEIGH DAN UR-LOS

Putu Agus Kartika Adhi Pratama
NRP 07111640000147

Dosen Pembimbing
Dr. Ir. Puji Handayani, M.T.
Prof. Ir. Gamantyo Hendrantoro, M.Eng.,Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT – EE 184801

PERFORMANCE ANALYSIS OF POINT-TO-POINT LARGE-SCALE ARRAY MIMO SYSTEM BASED ON HYBRID BEAMFORMING ON RAYLEIGH AND UR-LOS CHANNELS

Putu Agus Kartika Adhi Pratama
NRP 07111640000147

Supervisor(s)
Dr. Ir. Puji Handayani, M.T.
Prof. Ir. Gamantyo Hendrantoro, M.Eng.,Ph.D.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Analisis Kinerja Sistem *Point-to-Point Large-Scale Array* MIMO berbasis *Hybrid Beamforming* pada kanal *Rayleigh* dan UR-LoS**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Juni 2020

Putu Agus Kartika Adhi Pratama
NRP. 0711 16 4000 0147

ANALISIS KINERJA SISTEM POINT-TO-POINT LARGE-SCALE ARRAY MIMO BERBASIS HYBRID BEAMFORMING PADA KANAL RAYLEIGH DAN UR-LOS

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Telekomunikasi Multimedia
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I

Dr. Ir. Puji Handayani, M.T.
NIP. 196605101992032002

SURABAYA
JUNI, 2020

ANALISIS KINERJA SISTEM POINT-TO-POINT LARGE-SCALE ARRAY MIMO BERBASIS HYBRID BEAMFORMING PADA KANAL RAYLEIGH DAN UR-LOS

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Telekomunikasi Multimedia
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing II

Prof. Ir. Gamantyo Hendrantoro, M.Eng.,Ph.D.
NIP. 197011111993031002

SURABAYA
JUNI, 2020

ANALISIS KINERJA SISTEM *POINT-TO-POINT LARGE-SCALE ARRAY MIMO* BERBASIS *HYBRID BEAMFORMING* PADA KANAL *RAYLEIGH* DAN UR-LOS

Nama : Putu Agus Kartika Adhi Pratama
Pembimbing I : Dr. Ir. Puji Handayani, M.T.
Pembimbing II : Prof. Ir. Gamantyo Hendrantoro,M.Eng.,Ph.D.

ABSTRAK

Teknologi 5G yang sedang dikembangkan akan menggunakan antena berbasis MIMO untuk memaksimalkan *throughput* pada pengirim. Antena MIMO mengimplementasikan metode *Hybrid Beamforming* yang merupakan gabungan antara sistem *analog beamforming* dengan *digital beamforming* yang bertujuan membuat sistem dengan kompleksitas yang rendah serta mencapai hasil yang hampir sama dengan menggunakan sistem *full-digital beamforming*. Pada tugas akhir ini, akan dilakukan perbandingan kinerja sistem point-to-point MIMO pada kanal *independent Rayleigh fading* dan kanal *uniformly random Line-of-Sight* (URLoS) menggunakan Hybrid Beamforming. Grafik *spectral efficiency* terhadap SNR pada dua kanal yang berbeda didapatkan melalui simulasi pada MATLAB sehingga dapat dilakukan analisis pengaruh kanal *independent Rayleigh fading* dan URLoS pada sistem.

Dari hasil simulasi, dapat diketahui bahwa sistem *point-to-point large-scale array* MIMO pada kanal URLoS memiliki kinerja *spectral efficiency* yang lebih baik dibandingkan dengan kanal IRF. Terdapat peningkatan kinerja *spectral efficiency* sebesar 45% pada kanal URLoS dibandingkan dengan sistem yang berjalan pada kanal IRF. Perbedaan kinerja tersebut disebabkan karena pada kanal IRF tidak terjadi korelasi kanal, sedangkan pada kanal URLoS, antena yang berdekatan akan mengalami korelasi kanal sehingga dapat menyebabkan interferensi sinyal.

Kata kunci : Hybrid Beamforming, Point-to-Point Large-Scale Array MIMO, Independent Rayleigh Fading, UR-LoS, Spectral Efficiency, Signal-to-Noise Ratio (SNR)

[Halaman Ini Sengaja Dikosongkan]

PERFORMANCE ANALYSIS OF POINT-TO-POINT LARGE - SCALE ARRAY MIMO SYSTEM BASED ON HYBRID BEAMFORMING ON RAYLEIGH AND UR-LOS CHANNELS

Nama	**: Putu Agus Kartika Adhi Pratama**
Supervisor I	**: Dr. Ir. Puji Handayani, M.T.**
Supervisor II	**: Prof. Ir. Gamantyo Hendrantoro,M.Eng.,Ph.D.**

ABSTRACT

The 5G technology that is being developed will use MIMO-based antennas to maximize the throughput of the sender. The MIMO antenna will implement the Hybrid Beamforming method which is a combination of analog beamforming and digital beamforming systems that aim to create systems with low complexity and achieve almost the same results using a full-digital beamforming system. In this final project, we will compare the performance of point-to-point large-scale array MIMO systems on independent Rayleigh fading channels and uniformly random channels Line-of-Sight (URLoS) using Hybrid Beamforming. Spectral efficiency versus SNR graphs on two different channels are obtained through simulations on MATLAB so that analysis of independent Rayleigh fading and URLoS channels effects on the systems can be analyzed.

From the simulation results, it can be seen that the point-to-point large-scale array MIMO system on the URLoS channel has better spectral efficiency performance compared to the IRF channel. There is a 45% increase in spectral efficiency performance on URLoS channels compared to systems running on IRF channels. The difference in performance between those system due to the fact that there is no channel correlation in the IRF channel, while in the URLoS channel, adjacent antennas will experience channel correlation so that it can cause signal interference.

Keyword : Hybrid Beamforming, point-to-point large-scale array MIMO, Independent Rayleigh Fading, UR-LoS, Spectral Efficiency, Signal-to-Noise Ratio (SNR)

[Halaman Ini Sengaja Dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan yang Maha Esa atas segala rahmat, karunia, dan petunjuk yang telah dilimpahkan-Nya sehingga penulis mampu menyelesaikan Tugas Akhir dengan judul "**Analisis Kinerja Sistem *Point-to-Point Large-Scale Array* MIMO berbasis *Hybrid Beamforming* pada kanal *Rayleigh* dan UR-LoS**".

Tugas Akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan jenjang Pendidikan S1 pada Bidang Studi Telekomunikasi Multimedia, Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember. Atas selesainya penyusunan Tugas Akhir ini, penulis mengucapkan terima kasih sebesar-besarnya kepada:

1. Tuhan yang Maha Esa atas limpahan rahmat, karunia, dan petunjuk-Nya.
2. Ibu Ni Nyoman Sunantri dan Bapak I Nyoman Sueca selaku orang tua penulis atas doa dan cinta yang tak henti pada penulis dalam keadaan apapun. Semoga Tuhan yang Maha Esa senantiasa melindungi mereka.
3. Ibu Dr. Ir. Puji Handayani, M.T. dan Bapak Prof. Dr. Ir. Gamantyo Hendrantoro, Ph.D. selaku dosen pembimbing yang telah memberikan arahan, bimbingan dan perhatiannya selama proses penyelesaian Tugas Akhir ini.
4. Seluruh dosen dan karyawan Departemen Teknik Elektro ITS yang telah memberikan banyak ilmu dan menciptakan suasana belajar yang luar biasa.
5. Teman-teman seperjuangan e56 yang telah menemani dan memberikan dukungan selama masa kuliah sampai penyusunan Tugas Akhir ini.
6. Made Puspa Wedhanti yang selalu menemani saya selama proses perkuliahan dan penyusunan Tugas Akhir ini.
7. Dicky Rahmadi Prasetya selaku tim "*Hybrid Beamforming*".

Semoga Tugas Akhir ini dapat memberikan manfaat yang luas.

Surabaya, Juni 2020

Penulis

[Halaman Ini Sengaja Dikosongkan]

DAFTAR ISI

Halaman

DAFTAR GAMBAR

[Halaman Ini Sengaja Dikosongkan]

DAFTAR TABEL

[Halaman Ini Sengaja Dikosongkan]

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Saat ini perkembangan teknologi jaringan sudah memasuki "*Fifth Generation*" atau 5G, yang mempunyai tujuan untuk mencapai kecepatan transmisi data yang lebih tinggi, latensi jaringan yang lebih rendah, dan transmisi data yang lebih efisien. *Bandwidth* yang lebih besar jelas dibutuhkan untuk mencapai kecepatan transmisi data yang lebih tinggi. *Bandwidth* yang lebih besar ini di dapatkan dengan meningkatkan rentang frekuensi yang digunakan . Untuk teknologi 5G, terdapat tiga pita frekuensi yang dominan digunakan, yaitu 700 MHz *(Low band*), 3300 MHz (*Mid Band*), dan 24,3 – 29.2 GHz (*High Band*) [1]. Frekuensi lebih tinggi akan menyebabkan panjang gelombang yang lebih kecil atau *millimetre wave* (mmWave), oleh karena itu akan didapatkan elemen antena pada sistem dalam bentuk yang lebih kecil.

Kebutuhan teknologi 5G yang mencakup efisiensi spektral dan efisiensi energi yang tinggi dapat dicapai menggunakan sistem *Multiple-Input Multiple Output* (MIMO), sistem MIMO dengan banyak antena pada *precoder* maupun *detector* [2]. Pada umumnya, sinyal yang ditransmisikan oleh pemancar maupun diterima oleh penerima merupakan sinyal *omnidirectional*, sehingga seluruh sinyal akan ditransmisikan dengan energi yang sama [3]. Oleh karena itu, dibutuhkan teknik *beamforming* untuk memfokuskan sinyal yang diterima maupun dikirimkan menuju arah yang diinginkan untuk mengatasi *path loss* yang tidak diinginkan yang merupakan salah satu tantangan dalam penggunaan frekuensi mmWave [4]. Pembentukan *beamforming* tersebut dapat dicapai dengan menggunakan digital dan analog *precoding*, dimana *digital beamforming* berfungsi dengan menggunakan digital precoding yang mengalikan koefisien khusus untuk baseband yang sudah termodulasi pada RF *chain*, sedangkan pada analog beamforming terdapat *phase shifters* yang berguna untuk mengarahkan sinyal dan membatalkan sinyal yang mengganggu pada domain analog dan meminimalisir jumlah *analog-to-digital converter* (A/C). Pada umumnya, metode yang digunakan adalah *full-digital beamforming* digunakan, namun RF *chain* langsung di kombinasikan dengan elemen antena sehingga masing masing antena harus memilik RF *chain* masing masing. Hal tersebut mengakibatkan biaya yang lebih mahal dan konsumsi daya yang lebih tinggi jika diimplementasikan pada sistem *massive* MIMO. Maka dari itu

digunakanlah sistem *hybrid beamforming* [4.2].

Penelitian Tugas Akhir ini bertujuan untuk melihat kinerja sistem *point-to-point* LSA-MIMO pada dua kanal yang berbeda, yaitu IRF dimana terdapat hamburan isotropis antara *base station* (BS) dan *user equipment* (UE), sehingga tidak terdapat lintasan langsung antara BS dan UE. Kondisi tersebut dapat terjadi di kota-kota besar dengan gedung-gedung tinggi yang berdekatan sehingga agak sulit untuk memiliki lintasan langsung antara BS dengan UE. Kanal kedua yang disimulasikan yaitu URLoS, dimana terdapat lintasan langsung antara UE dan BS. Kondisi tersebut juga dapat terjadi pada kondisi nyata dimana pada kota-kota yang tidak memiliki banyak bangunan tinggi, sehingga BS dapat melihat secara langsung atau memiliki lintasan langsung menuju UE. Efisiensi spektral dipilih sebagai parameter utama yang diukur ada penelitian ini dikarenakan pada kondisi nyata, operator telekomunikasi memiliki *bandwidth* yang terbatas. Semakin tinggi trafik komunikasi, maka dibutuhkan kapasitas kanal yang lebih tinggi. Salah satu cara untuk mencapai hal tersebut adalah dengan meningkatkan *bandwidth,* namun hal tersebut sulit dicapai karena operator telekomunikasi tidak mempunyai *bandwidth* yang tidak terbatas. Oleh karena itu peneliti ingin melihat bagaimana kedua jenis kanal yang berbeda mempengaruhi sistem *point-to-point* LSA-MIMO jika dibandingkan satu sama lain dengan melihat sisi efisiensi spektral terhadap SNR.

1.2 Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana pengaruh jumlah RF *beamformer* terhadap kinerja sistem *point-to-point* LSA-MIMO?
2. Bagaimana kinerja *spectral efficiency versus* (SNR) *hybrid beamforming point-to-point* LSA-MIMO pada kanal *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (UR-LoS)?

1.3 Tujuan

Tujuan yang diharapkan dapat tercapai setelah selesainya tugas akhir ini adalah sebagai berikut :

1. Mengetahui kinerja teknik *Hybrid Beamforming* menggunakan *large-scale antenna arrays* pada *point-to-point* LSA-MIMO.

2. Menguji kinerja *spectral efficiency* sistem *Hybrid Beamforming point-to-point* LSA-MIMO pada kanal *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (UR-LoS).

1.4 Batasan Masalah

Untuk menyelesaikan permasalahan yang terdapat pada tugas akhir ini, terdapat beberapa hal yang akan dilakukan dalam penelitian ini yaitu sebagai berikut :

1. *Hybrid* (analog dan digital) *beamforming*
2. Kanal *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (UR-LoS)
3. Parameter yang dianalisa adalah *spectral efficiency versus signal-to-noise ratio* (SNR)
4. Sistem *point-to-point large scale array* MIMO
5. Modulasi BPSK
6. Simulasi dilakukan menggunakan *software* MATLAB.

1.5 Metodologi Penelitian

Langkah-langkah yang akan digunakan dalam penelitian Tugas Akhir ini adalah sebagai berikut :

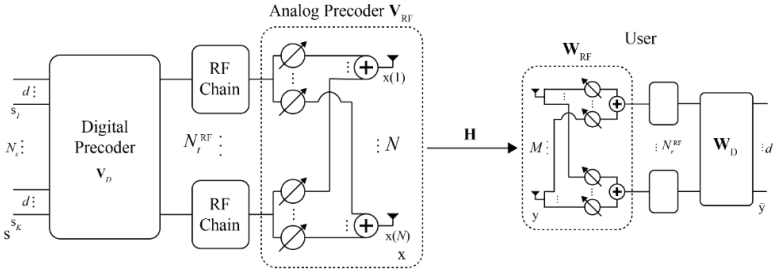
1. Studi Pustaka

Studi Literatur dilakukan untuk mencari dan mengumpulkan literatur dan kajian yang berkaitan dengan masalah-masalah yang ada pada Tugas Akhir ini, baik berupa buku referensi, jurnal, dan sumber-sumber lain yang berhubungan dengan:

a) Prinsip *hybrid beamforming*
b) Prinsip *sistem point-to-point* LSA-MIMO
c) Prinsip *fan beam array*
d) *Independent Rayleigh fading*
e) *Uniformly random Line-of-Sight (UR-LoS)*
f) Modulasi BPSK

2. Pemodelan Sistem *Point-to-Point* LSA-MIMO menggunakan *Hybrid Beamforming*

Sistem yang akan digunakan merupakan *point-to-point* LSA-MIMO dengan metode *hybrid beamforming*, oleh karena itu perlu dilakukan pengkajian sistem sebagai langkah awal penelitian Tugas Akhir ini. Pengkajian sistem dilakukan menggunakan simulasi software MATLAB.



Gambar 1.1 Blok diagram point-to-point LSA-MIMO berbasis HB

3. Simulasi pada Kanal *Independent Rayleigh Fading* dan UR-LoS

Hasil dari simulasi sistem *point-to-point* LSA-MIMO kemudian dilakukan pengujian menggunakan dua kanal yang berbeda yaitu *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (UR-LoS) guna menguji dan meneliti hasil yang akan didapatkan.

4. Analisis Kinerja terhadap SE dan SNR

Setelah mendapatkan hasil simulasi pada dua kanal, maka akan dijadikan acuan untuk kemudian dilakukan analisa terhadap parameter *spectral efficiency* (SE) dan *signal-to-noise ratio* (SNR).

5. Analisis Data dan Penarikan Kesimpulan

Selanjutnya setelah melakukan analisa terhadap data yang sudah didapatkan, dilakukan penarikan kesimpulan mengenai hasil kinerja pengujian sistem *point-to-point* LSA-MIMO dengan metode *hybrid beamforming* dan hasil perbandingan sistem pada kanal *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (UR-LoS).

1.6 Sistematika Penulisan

Penulisan laporan tugas akhir ini mempunyai sistematika penulisan yang berisi tentang penjelasan mengenai tahapan-tahapan yang dilakukan dalam penelitian tugas akhir ini. Berikut penjelasan mengenai tahapan-tahapan yang digunakan dalam penulisan tugas akhir :

- **BAB 1 PENDAHULUAN**

  Bab pendahuluan berisi mengenai latar belakang, pepersamaanan masalah, tujuan penelitian, metodologi penelitian, dan sistematika penulisan.

- **BAB 2 TINJAUAN PUSTAKA**

  Pada bab tinjauan pustaka membahas mengenai teknologi 5G, *beamforming,* kanal propagasi, *point-to-point* LSA-MIMO, dan estimasi BER.

- **BAB 3 PEMODELAN DAN SIMULASI SISTEM**

  Pada bab tiga menjelaskan mengenai metodologi penelitian dalam merancang sistem *point-to-point* LSA-MIMO pada kanal *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (UR-LoS).

- **BAB 4 PENGAMBILAN DAN ANALISA DATA**

  Pada bab empat menjelaskan mengenai hasil pengujian berdasarkan simulasi sistem *point-to-point* LSA-MIMO terhadap dengan parameter *spectral efficiency versus signal-to-noise ratio* (SNR) menggunakan kanal *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (UR-LoS).

- **BAB 5 KESIMPULAN DAN SARAN**

  Pada bab 5 ini berisi mengenai kesimpulan dan saran penulis terkait dengan penelitian tugas akhir yang telah dilakukan.

1.7 Relevansi

Hasil dari penelitian tugas akhir ini diharapkan dapat memberi manfaat sebagai berikut :

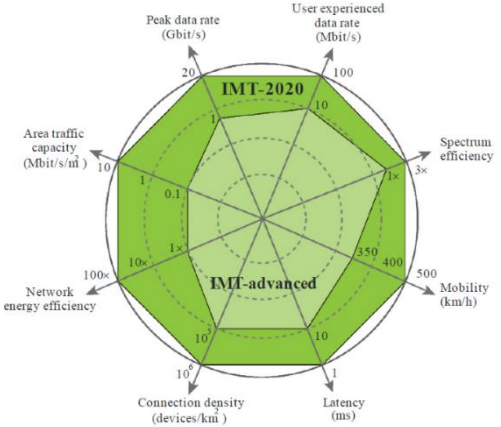
1. Menunjukkan hasil kinerja sistem *point-to-point* LSA-MIMO dengan membandingkan kinerja kanal *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (UR-LoS).
2. Memberikan solusi mengenai nilai parameter *spectral efficiency versus signal-to-noise ratio* (SNR) terkait dengan pengaruh jumlah RF *beamformer.*

[Halaman Ini Sengaja Dikosongkan]

BAB 2
TINJAUAN PUSTAKA

2.1 Teknologi 5G

Perkembangan teknologi jaringan nirkabel yang semakin pesat didukung oleh pengaruhnya terhadap pengembangan pada bidang sosial dan ekonomi. Kondisi teknologi saat ini yang membutuhkan high-level-services, mendorong pengembangan generasi baru jaringan nirkabel, yaitu 5G. Beberapa hal yang diharapkan dapat dilakukan oleh teknologi 5G tersebut adalah sistem dengan kapasitas yang lebih tinggi, data rate yang lebih tinggi, konektifitas perangkat yang masif, latensi yang lebih singkat, hemat energi, dan biaya yang lebih rendah [5]. Lebih spesifik dijelaskan dalam pengembangan IMT-2020 yang menyebutkan delapan parameter untuk memenuhi key capabilities (KCs) yang ditunjukkan dalam Gambar 2.1 [6].



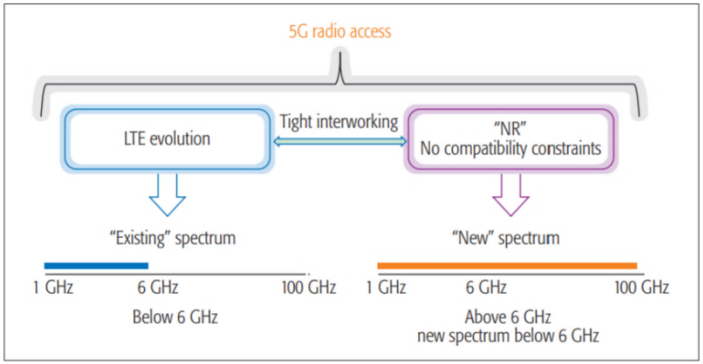
Gambar 2.1 Perbandingan KCs teknologi 4G dan 5G [6].

Secara khusus, pengguna layanan 5G dapat diklasifikasikan menjadi eMBB (Enhanced mobile broadband), URLLC (Ultra-reliable low-latency communications), dan mMTC (Massive machine type communications) [7]. Layanan 5G dalam aplikasi tersebut diharapkan dapat mencapai data rate 20 Gbps dalam kondisi dan skema tertentu, dengan data rate masing-masing pengguna mencapai 100 Mbps dalam area urban maupun sub-urban dan 1 Gbps untuk dalam ruangan, serta area

kapasitas trafik mencapai 10 Mbps/m2 [6]. Teknologi 5G diperkirakan akan memakai frekuensi diatas 6 GHz dikarenakan bandwidth yang tersedia lebih lebar, yaitu lebih besar dari 500 MHz [8].

Frekuensi angtara 30 GHz sampai 300 GHz merupakan frekuensi yang masih diteliti dalam penggunaan teknologi 5G [9]. Di Indonesia sendiri, rentang frekuensi yang mungkin digunakan adalah 24,3 GHz sampai 29,2 GHz. Rentang frekuensi tersebut termasuk ke dalam millimeter Wave (mmWave). Perkembangan teknologi komponen semikonduktor pada masa kini semakin mendukung digunakannya rentang frekuensi mmWave untuk mencapai trafik data yang lebih tinggi dengan biaya produksi yang lebih rendah dan konsumsi daya yang lebih rendah pula. Namun, semakin tinggi frekuensi yang digunakan maka gangguan akibat uap air, hujan dan halangan akibat pohon maupun gedung akan semakin mengganggu jalannya transmisi maupun penerimaan sinyal.

Frekuensi yang akan digunakan merupakan salah satu kandidat frekuensi 5G di Indonesia. Frekuensi tersebut akan lebih tinggi dibandingkan dengan frekuensi yang digunakan pada teknologi sebelumnya, yaitu 4G. Hal ini memungkinkan ukuran elemen antena yang lebih kecil sehingga dapat digunakan antena jamak atau antena array, namun salah satu kekurangannya adalah pada frekuensi tinggi ini akan lebih sensitive terhadap redaman maupun noise.



Gambar 2.2 Akses frekuensi radio untuk teknologi 5G [7]

2.2 Modulasi *Binary Phase Shift Keying* (BPSK)

Pada modulasi *binary phase shift keying* (BPSK), digunakan dua skema sinyal yang berbeda, yaitu bit 1 dan 0 yang berbeda fase sebesar 180°. Sinyal BPSK dengan biner 1 didapatkan dengan persamaan berikut[19]:

$$s_{BPSK}(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t + \theta_c) \qquad 0 \le t \le T_b \qquad (2.1)$$

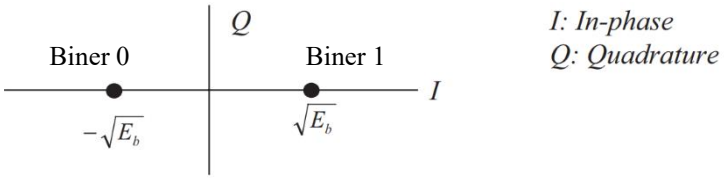
Dimana E_b adalah energi per bit dan T_b adalah periode per bit. Sinyal BPSK dengan biner 0 didapatkan dengan persamaan berikut:

$$s_{BPSK}(t) = -\sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t + \theta_c) \qquad 0 \le t \le T_b \qquad (2.2)$$

Dengan menggunakan basis sinyal tersebut, maka sinyal BPSK dapat direpresentasikan sebagai berikut:

$$s_{BPSK} = m(t)\sqrt{\frac{2E_b}{T_b}\cos(2\pi f_c t + \theta_c)} \qquad (2.3)$$

Sinyal BPSK dapat dilihat secara geometris pada Gambar 2.3 yang disebut sebagai diagram konstelasi yang merepresentasikan energi symbol yang mungkin terjadi.



Gambar 2.3 Diagram konstelasi modulasi BPSK [19]

Dengan mengasumsikan sebuah pulsa *rectangular* $(p(t) = rect\big((t - T_b/2)T_b\big)$, persamaan sinyal $s_1(t)$ dan $s_2(t)$ dengan modulasi BPSK sebagai berikut:

$$s_1(t) = \sqrt{\frac{2E_b}{T_b}\cos(2\pi f_c t)} \qquad (2.4)$$

$$s_2(t) = -\sqrt{\frac{2E_b}{T_b}\cos(2\pi f_c t)} \tag{2.5}$$

Modulasi BPSK mempunyai basis sinyal seperti berikut:

$$\varphi_1(t) = -\sqrt{\frac{2}{T_b}\cos(2\pi f_c t)} \quad 0 \le t \le T_b \tag{2.6}$$

Dengan menggunakan basis sinyal tersebut, maka sinyal BPSK dapat direpresentasikan sebagai berikut:

$$s_{BPSK} = \left\{\sqrt{E_b}\varphi_1(t), -\sqrt{E_b}\varphi_1(t)\right\} \tag{2.7}$$

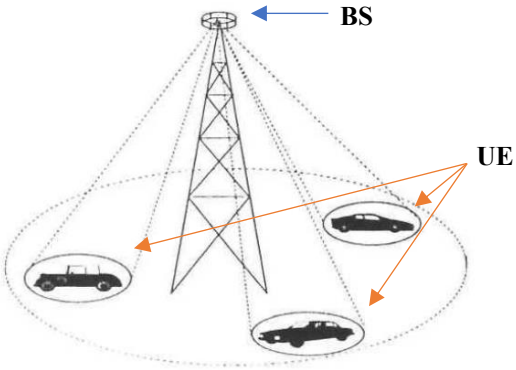
2.3 Beamforming

Beamforming merupakan teknik pengarahan atau pembentukan beam sinyal informasi dengan menggunakan antena jamak (*array*). Teknik ini memfokuskan energi sinyal menuju arah yang diinginkan sehingga dapat mentransmisikan maupun menerima pancaran sinyal lebih kuat ke arah tertentu dibandingkan dengan memancarkan energi ke segala arah seperti teknik pada umumnya. *Beamforming* dapat digunakan pada kedua sisi sistem komunikasi, yaitu pada *transmitter* (pemancar) dan *receiver* (penerima) untuk mencapai selektivitas spasial yang berguna untuk mengarahkan sinyal sekaligus menghilangkan pancaran menuju arah yang tidak diinginkan.

Beamforming terbentuk dari suatu *beamfomer*, sebuah perangkat atau komponen yang berguna untuk memperkirakan sinyal yang datang dari arah yang diinginkan dengan adanya gangguan dan sinyal yang menggangu dengan menggunakan *spatial filtering* untuk memisahkan beberapa sinyal yang tumpang tindih namun berasal dari lokasi yang berbeda [9]. Sehingga *beamforming* merupakan suatu upaya membentuk pancaran (*beam*) sinyal dengan memfokuskan daya radiasi untuk mengirimkan maupun menerima sinyal dari arah yang diinginkan.

Salah satu solusi untuk mengurangi gangguan pada sinyal yang menggunakan frekuensi mmWave adalah dengan menggunakan antena jamak yang digabungkan dengan teknik *beamforming* [9]. Penggunaan lebih dari satu antena atau antena jamak (*array antenna systems*) dapat mendukung teknik *beamforming* untuk mendapatkan daya yang lebih

besar sehingga dapat mencapai jarak pancaran yang lebih jauh. Hal ini juga didukung oleh ukuran elemen antena yang lebih kecil jika menggunakan rentang frekuensi mmWave, sehingga dapat memuat lebih banyak antena pada suatu dimensi tertentu dibandingkan dengan ukuran antena yang digunakan pada teknologi 4G. Dengan adanya teknik beamforming dan antena jamak, maka daya sinyal, arah pancaran sinyal, serta arah penerimaan sinyal atau direction of arrival (DOA) dapat dikontrol (*beamsteering*) sehingga dapat mengurangi konsumsi daya pada arah yang tidak dituju. Terdapat dua model *beamforming,* yaitu *analog beamforming* dan *digital beamforming*.



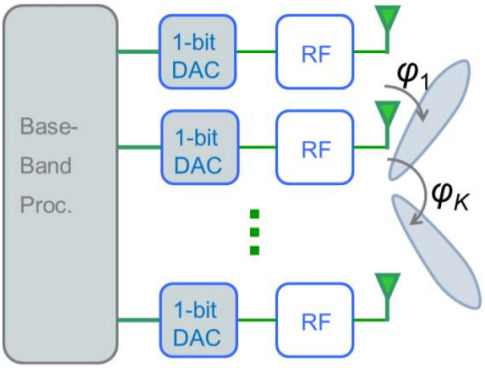
Gambar 2.4 Komunikasi UE dan BS dengan *beamforming* [13].

2.3.1 Digital Beamforming

Pada awalnya, *digital beamforming* diciptakan untuk aplikasi pada sistem sonar dan radar. Pada sistem *digital beamforming*, sinyal yang dideteksi dan diterima akan malalui proses digitalisasi dengan mengubah informasi sinyal *radio frequency* (RF) pada setiap elemen antena ke dalam dua urutan sinyal *baseband* biner, termasuk amplitude dan fase sinyal yang diterima pada setiap elemen antena *array. Beamforming* dilakukan secara digital menggunakan prosesor sinyal maupun *beamforming chip* yang tertanam, dengan melakukan pembobotan pada sinyal-sinyal digital tersebut, dengan mengatur amplitudo dan fase sinyal sedemikian rupa sehingga ketika sinyal tersebut digabungkan, akan membentuk beam yang diinginkan [9,10].

Beberapa bentuk *digital beamforming* yaitu estimasi *direction of arrival* (DOA), program pengontrol pola radiasi antena, dan *adaptive steering* pada beam sinyal dan nulls untuk meningkatkan *signal-to-interference noise ratio* (SINR) [12]. Berikut merupakan beberapa kelebihan digital beamforming dibandingkan dengan sistem *phased arrays* konvensional:

1. *Steered high-gain beams* dalam jumlah besar dapat dibentuk tanpa mengakibatkan penurunan pada *signal-to-noise ratio* (SNR).
2. *User* dapat mempunyai beam-nya masing-masing, dengan demikian user mendapatkan gain maksimum.
3. Prosesor sinyal dapat mengoptimalkan kinerja sistem saat semua informasi diterima oleh antena array.
4. Parameter amplitudo dan fase antara *transceiver* dapat diperbaiki secara real-time karena sistem *digital beamforming* dapat melakukan kalibrasi sistem antena dalam domain digital secara *real-time*.
5. Dapat membentuk *adaptive beamforming* untuk mengurangi efek *multipath fading*.



Gambar 2.5 Blok diagram *digital beamforming* pada pemancar [11].

Akurasi pengubahan sinyal dari sinyal analog menjadi sinyal digital menjadi kunci dalam pengaplikasian *digital beamforming*, dengan menggunakan *complete heterodyne receivers* [9].

Hasil sinyal dengan noise menggunakan elemen antenna ke-*k* pada sistem digital beamforming pada waktu t dinyatakan dalam[10]:

$$X_k(n) = u_k(n) + iv_k(n) \tag{2.8}$$

dimana $u_k(n)$ dan $v_k(n)$ merupakan bentuk real dan imajiner dari bagian kompleks sinyal input baseband. Pembobotan $W_k(n)$ untuk elemen antena ke-k menggunakan algoritma adaptif ditunjukkan dalam[10]:

$$W_k(n) = a_k(n)e^{i\theta_k(n)} = a_k(n)[\cos(\theta_k(n)) + i\sin(\theta_k(n))] \tag{2.9}$$

dimana $a_k(n)$ dan $\theta_k(n)$ adalah factor amplitude dan pergeseran fase pada waktu t. Sinyal output dari antena ke-k ditunjukkan dalam[10]:

$$\begin{aligned} X_k(n)W_k(n) = a_k\{&[u_k(n)\cos(\theta_k) - v_k(n)\sin(\theta_k)] \\ &+ j[u_k(n)\sin(\theta_k) + v_k(n)\cos(\theta_k)]\} \end{aligned} \tag{2.10}$$

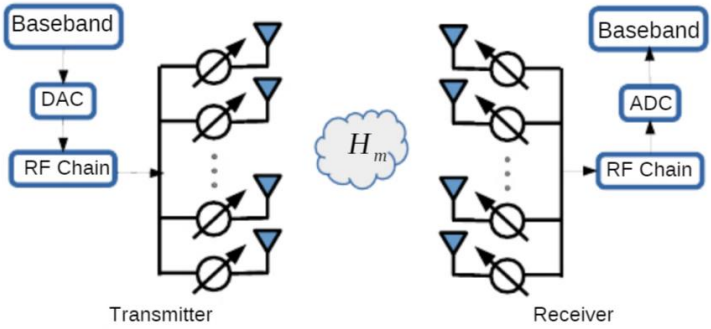
Jika nilai error yang diperoleh pada pengecekan setiap waktu berada diatas nilai threshold, maka fungsi pembobotan diperbarui menggunakan algoritma adaptif pada [10]. Output yang diinginkan didapat saat error dikurangi untuk mencapai level threshold.

2.3.2 Analog Beamforming

Sistem *analog beamforming* dengan antena yang menggunakan matriks *hybrid* dan *fixed phase shifters* telah ditemukan lebih dari 50 tahun lalu. Fase sinyal hasil sistem *analog beamforming* yang ditransmisikan dapat dikontrol menggunakan *phase shifters* dengan harga yang murah dan juga menggunakan RF *switch* untuk membentuk sudut *beam* pada sinyal. Pemodelan *analog beamforming* yang diaplikasikan pada sistem MIMO dengan *phase shift networks* yang berguna untuk mengurangi interferensi sinyal dan meminimalisir penggunaan ADC maupun DAC diteliti dalam [14], namun masih sulit untuk mengontrol arah sinyal yang tidak diinginkan pada pemodelan tersebut [2].

Analog beamforming memfokuskan dan mengarahkan energi yang dipancarkan menuju arah tertentu sehingga antena dapat menerima maupun mengirimkan sinyal yang diinginkan. Lensa *microwave*, *waveguide*, saluran transmisi, cetakan sirkuit *microwave*, dan *hybrid* dapat digunakan sebagai *beamforming networks*. Sebagai contohnya pada sistem antena *parabolic*, kabel antena mendapatkan energi sinyal yang telah ditangkap oleh piringan antena parabola dengan *aperture* antena yang sudah dibentuk oleh perimeter antena parabola, sehingga piringan dan kabel antena disebut sebagai integrator spasial. Energi yang didapat piringan antena berasal dari sumber medan jauh dengan asumsi sejajar

dengan arah yang diinginkan antena. Oleh karena itu, *analog beamforming* disebut sebagai penyaringan spasial. Penyaringan spasial pada analog beamforming umumnya menggunakan antena *array* untuk mengatur amplitude dan fase sinyal guna membentuk *beam* yang diinginkan [9].



Gambar 2.6 Blok diagram *analog beamforming* [15].

2.3.3 Hybrid Beamforming

Teknik *hybrid beamforming* mencoba untuk mengimbangkan teknik *analog beamforming* dengan teknik *digital beamforming*. *Analog beamforming* dengan *phase shifter* yang murah namun output sistem *full-digital beamforming* lebih baik daripada *analog beamforming* karena amplitude dari *phase shifters* yang tidak fleksibel. Pada *hybrid beamforming*, teknik *digital beamforming* digunakan untuk membangkitkan sinyal *baseband*, sementara jumlah ADC/DAC dapat dikurangin dengan *analog beamforming* sehingga dapat memperbaiki daya output yang dihasilkan dan kompleksitas sistem yang lebih rendah.

2.4 Kanal Propagasi

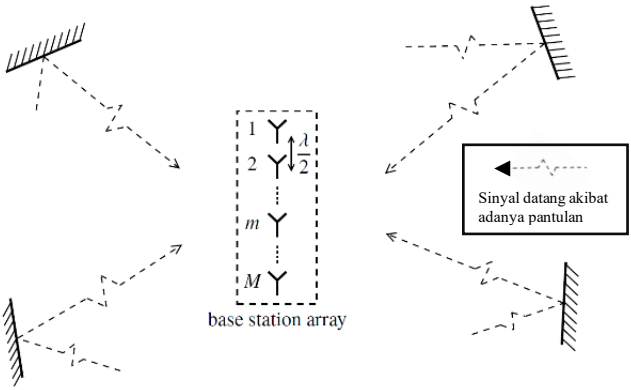
Kanal propagasi merupakan sebuah saluran atau lintasan yang harus dilalui jika transmisi data ingin dilakukan antara pemancar dan penerima. Pada kanal propagasi, sinyal informasi yang dikirimkan dapat mengalami penguatan atau pelemahan yang dapat berupa refleksi, difraksi, dan hamburan. Hal tersebut dapat mengakibatkan sinyal informasi mengalami penundaan atau delay dan perbedaan arah kedatangan dari pantulan-pantulan sinyal sehingga terjadi fluktuasi daya sinyal yang dapat menjumlahkan daya sinyal pada sisi penerima. Selain itu, amplitudo

maupun frekuensi carrier sinyal informasi dapat bergeser atau berubah akibat adanya *noise* pada kanal [16]. Karena adanya gangguan-gangguan tersebut pada sinyal informasi, maka pada sisi penerima perlu dilakukan rekonstruksi sinyal. Model kanal propagasi pada suatu sistem komunikasi pada umumnya berupa AWGN maupun kanal *multipath fading*.

Terdapat beberapa model kanal *favorable propagation* yaitu *independent Rayleigh fading* (IRF) dan *uniformly random line-of-sight* (URLoS). Matriks *G* merepresentasikan kanal yang terdistribusi acak akibat adanya hamburan, Dalam model kanal ini, diamati distribusi dari nilai-nilai singular σ_k pada matriks *G*, atau probabilitas nilai ekstrim $\sigma_{max}/\sigma_{min}$ dari $\{\sigma_k\}$, maupun probabilitas ratio antara jumlah kapasitas C_{sum} dan batas atas, Δ_c dan jumlah kapasitas yang dibutuhkan oleh *G* untuk mencapai batas atas, $\Delta_{\rho_{ul}}$, melewati batas yang ditentukan. Secara khusus, dua kasus kanal yang berbeda secara fisik akan digunakan sebagai perbandingan yaitu *independent Rayleigh fading* dan UR-LoS dan ditentukan sel tunggal dengan jarak antena $\lambda/2$ serta BS dengan antena jamak yang uniform dan linear [17].

2.4.1 Independent Rayleigh Fading

Sistem akan beroperasi pada lingkungan dimana terdapat hamburan isotropic yang padat dengan mengasumsikan G sebagai elemen acak yang independen $\{g_k^m\}$ dengan reratan dan distribusi nol $CN(0, \beta_k)$ dan setiap antena mendapatkan banyak sinyal hamburan independen.



Gambar 2.7 Skema *Independent Rayleigh Fading* [16]

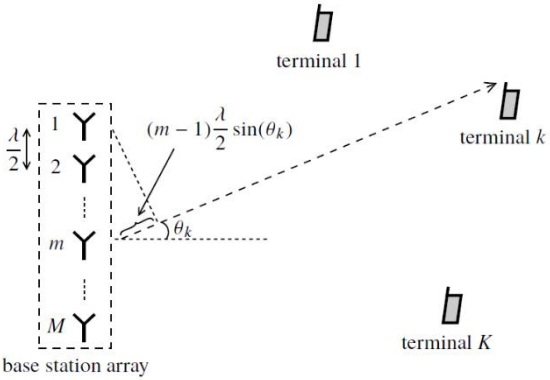
Dalam *independent Rayleigh Fading*, $E\{|g_k^m|^2\} = \beta_k$ dan $E\{g_k^{m*} g_{k'}^{m'}\} = 0$ pada $k' \neq k$ atau $m' \neq m$. Dengan hukum *large numbers*, maka:

$$\frac{1}{M}\left|\left|g_k\right|\right|^2 \rightarrow \beta_k, \quad M \rightarrow \infty, \qquad k = 1, \ldots, K, \tag{2.11}$$

$$\frac{1}{M} g_k^H g_{k'} \rightarrow 0 \qquad M \rightarrow \infty, \qquad k \neq k'. \tag{2.12}$$

2.4.2 Uniformly Random Line-of- Sight (UR-LoS)

Pada skema UR-LoS, sistem tidak mengalami hamburan dan semua saluran memiliki line-of-sight antara BS dan user dengan mengasumsikan elemen antena ke-*k* terletak pada medan jauh dengan sudut θ_k dan diukur dari *boresight* antena [17]. Sudut $\{\sin(\theta_k)\}$ didistribusikan secara seragam di interval [-1, 1] Vektor kanal g_k dimodelkan sebagai berikut:



Gambar 2.8 Skema *Uniformly Random Line-of-Sight* [16]

$$g_k = e^{i\phi_k}\left[1\ e^{-i2\pi\frac{d}{\lambda}sin(\theta_k)} \ldots\ e^{-i2\pi(M-1)\frac{d}{\lambda}sin(\theta_k)}\right]^T, \tag{2.13}$$

dimana ϕ_k merupakan angka acak antara $-\pi$ dan π yang terdistribusi

secara uniform. Phase shift terhubung pada jarak acak antar array dengan ke-k terminal. Untuk dua terminal k dan k' dengan sudut θ_k dan $\theta_{k'}$ dimana $\theta_k \neq \theta_{k'}$,

$$\frac{1}{M} g_k^H g_{k'} = \frac{1}{M} \sqrt{\beta_k \beta_{k'}} e^{-i\left(\phi_k - \phi_{k'}\right)} \sum_{m=0}^{M-1} e^{im\pi\left(\sin(\theta_k) - \sin\left(\theta_{k'}\right)\right)}$$

$$= \frac{1}{M} \sqrt{\beta_k \beta_{k'}} e^{-i\left(\phi_k - \phi_{k'}\right)} \frac{1 - e^{iM\pi\left(\sin(\theta_k) - \sin\left(\theta_{k'}\right)\right)}}{1 - e^{i\pi(\sin(\theta_k) - \sin(\theta_{k'}))}}$$

$$\rightarrow 0. \qquad M \rightarrow \infty, \tag{2.14}$$

$$\frac{1}{M} \left|\left| g_k \right|\right|^2 = \beta_k, \qquad k = 1, \ldots, K. \tag{2.15}$$

2.5 Point-to-Point LSA-MIMO

Sebuah BS dengan antena array yang melayani satu user dengan antena array disebut sebagai sistem *point-to-point large-scale array multiple-input multiple-output* (LSA-MIMO). Diasumsikan pada uplink dengan jumlah antena K pada BS dan jumlah antena M pada user. Maka model sinyal untuk *point-to-point* LSA-MIMO adalah sebagai berikut [17]:

$$\mathrm{y} = \sqrt{\rho} G x + w. \tag{2.16}$$

Vektor yang merepresentasikan sinyal yang ditransmisikan terdiri dari vector sinyal yang diterima $y = [y_1, \ldots, y_M]^T$, vektor sinyal yang ditransmisikan $x \triangleq [x_1, \ldots, x_k]^T$, *noise* yang terdistribusi nirmal kompleks(0, 1) $w = [w_1, \ldots, w_m]^T$, dan penguatan total:

$$G = \begin{bmatrix} g_1^1 & \cdots & g_K^1 \\ \vdots & \ddots & \vdots \\ g_1^M & \ldots & g_K^M \end{bmatrix} \tag{2.17}$$

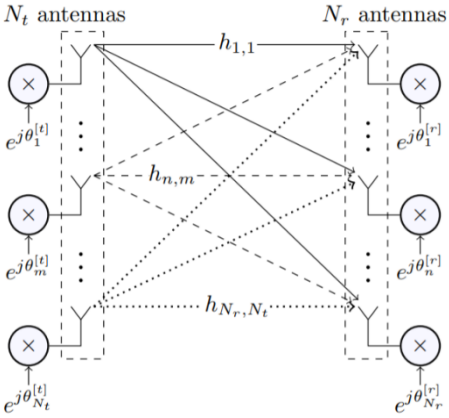
dengan g_K^M merupakan penguatan yang terjadi antara terminal dengan antena k dan BS dengan antena m. Kapasitas dari sistem MIMO tersebut adalah sebagai berikut [18]:

$$C = MkB \, log_2 \left(1 + \frac{S}{N}\right) \tag{2.18}$$

dimana M adalah jumlah antena transmisi, k adalah jumlah antena penerima, B adalah bandwidth dari sinyal, dan $\frac{S}{N}$ adalah signal-to-noise

ratio.

Sistem *point-to-point* LSA-MIMO akan digunakan sebagai sistem komunikasi yang digunakan dalam penelitian Tugas Akhir ini. Pada sistem tersebut, akan terdapat satu BS yang memiliki antena jamak dengan konfigurasi *uniform linear array* (ULA) di dalamnya serta satu user atau terminal yang juga memiliki antena dengan konfigurasi ULA di dalamnya. Hal ini bertujuan untuk menganalisa pengaruh kanal *independent Rayleigh fading* dan *uniformly random Line-of-Sight* (LoS) terhadap sistem *point-to-point* LSA-MIMO.



Gambar 2.9 Skema *Point-to-Point* MIMO [17]

2.5.1 Point-to-Point LSA-MIMO berbasis HB

Sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming* menggunakan jumlah RF *chain* setengah dari yang digunakan pada sistem dengan metode *full-digital beamforming.* Untuk merangkai sinyal yang akan ditransmisikan pemancar, hasil pemrosesan sinyal akan melalui $\mathbf{V}_{\mathrm{RF}}$. Secara metematis, sinyal yang ditransmisikan pemancar adalah sebagai berikut:

$$x = \mathrm{V_{RF}V_D s} = \sum_{l=1}^{K} \mathrm{V_{RF}V_D s} \tag{2.19}$$

Dimana $\mathrm{V_{RF}}$ merupakan matriks dengan dimensi $N\ x\ N_t^{RF}$, V_d matriks dengan dimensi $N_t^{RF}\ x\ N_s$, s adalah matriks kompleks dengan dimensi $N_s\ x\ 1$, dan l menunjukkan user saat itu. Sinyal yang ditransmisikan

pemancar, akan melewati kanal dengan asumsi kanal *independent Rayleigh fading* dan UR-LoS. Maka, sinyal yang diterima oleh user *K* adalah sebagai berikut:

$$y_k = \mathrm{H}_k \mathrm{V}_{\mathrm{RF}} \mathrm{V}_{\mathrm{D}_k} \mathrm{s}_k + \mathrm{H}_k \sum_{l \neq k} \mathrm{V}_{\mathrm{RF}} \mathrm{V}_{\mathrm{D}_l} \mathrm{s}_l + z_k \tag{2.20}$$

Dimana H_k adalah matriks penguatan kanal kompleks dengan dimensi *M x N* dan z_k adalah *noise* yang terdistribusi *Gaussian*. Selanjutnya, penerima memproses sinyal yang diterima dengan menggunakan RF *combiner,* W_{RF} , yang diimplementasikan dengan menggunakan penggeser fase sehingga didapatkan $\left|\mathrm{W}_{\mathrm{RF}_k}(i,j)\right|^2 = 1$. Hasil olahan sinyal tersebut, kemudian mengalami pengembalian frekuensi ke *baseband* dengan menggunakan RF *chain* pada sisi penerima, N_r^{RF} . Dengan menggunakan *digital combiner*, $\mathrm{W}_{\mathrm{D}_k}$, sinyal yang telah diproses pada penerima adalah sebagai berikut:

$$\tilde{y}_k = \mathrm{W}_{\mathrm{t}_k}^H \mathrm{H}_k \mathrm{V}_{\mathrm{t}_k} \mathrm{s}_k + \mathrm{W}_{\mathrm{t}_k}^H \mathrm{H}_k \sum_{l \neq k} \mathrm{V}_{\mathrm{t}_l} \mathrm{s}_l + \mathrm{W}_{\mathrm{t}_k}^H \mathrm{s}_l \tag{2.21}$$

Dimana $\mathrm{V}_{\mathrm{t}_k} = \mathrm{V}_{\mathrm{RF}} \mathrm{V}_{\mathrm{D}}$ dan $W_{\mathrm{t}_k} = \mathrm{W}_{\mathrm{RF}_k} \mathrm{W}_{\mathrm{D}_k}$.

Untuk memaksimalkan efisiensi spektral secara keseluruhan dengan total daya pancar yang terbatas, dibutuhkan *hybrid precoder* pada pemancar dan user yang optimal dengan memecahkan masalah berikut:

$$\mathrm{Tr}(\mathrm{V}_{\mathrm{RF}} \mathrm{V}_{\mathrm{D}} \mathrm{V}_{\mathrm{D}}^H \mathrm{V}_{\mathrm{RF}}^H) \leq P \tag{2.22a}$$

$$|\mathrm{V}_{\mathrm{RF}}(i,j)|^2 = 1, \ \forall i,j \tag{2.23b}$$

$$|\mathrm{W}_{\mathrm{RF}}(i,j)|^2 = 1, \ \forall i,j \tag{2.24c}$$

Dimana *P* adalah total daya yang dibutuhkan oleh pemancar.

Untuk membangkitkan *digital precoder*, dengan mengasumsikan V_{RF} tetap, kanal efektif didapatkan dengan persamaan $\mathrm{H}_{\mathrm{eff}} = \mathrm{HV}_{\mathrm{RF}}$ dan $\mathrm{Q} = V_{RF}^H V_{RF}$, sehingga didapatkan *digital precoder,* V_d, sebagai berikut:

$$\mathrm{V}_d = Q^{-1/2} U_e \Gamma_e \tag{2.25}$$

Dimana U_e adalah set dari *right singular vectors* dengan nilai singular

sebanyak N_s dari persamaan $H_{eff}Q^{-1/2}$ dan Γ_e adalah matriks diagonal dari daya yang dialokasikan untuk setiap aliran data, yang dapat dipersamaankan dengan $\Gamma_e \approx \sqrt{P/N^{RF}}I$, dengan I adalah matriks identitas.

Untuk memodelkan RF *precoder,* dengan mengasumsikan $V_D V_D^H \approx \gamma^2 I$, RF *precoder* didapatkan dengan memecahkan permasalahan sebagai berikut:

$$\max_{V_{RF}} \; log_2 \left| I \; + \frac{\gamma^2}{\sigma^2} V_{RF}^H F_1 V_{RF} \right| \tag{2.26}$$

Serta memenuhi persamaan 3.5b, dimana $F_1 = H^H H$. Untuk menjabarkan kontribusi $V_{RF}(i,j)$ pada persamaan 3.7, maka dapat ditulis ulang dengan persamaan berikut:

$$log_2 \left| C_j \right| + log_2 \left(2\, Re\left\{ V_{RF}^*(i,j)\eta_{ij} \right\} + \zeta_{ij} + 1 \right) \tag{2.27}$$

Dimana C_j, η_{ij}, dan ζ_{ij} dapat dipersamaankan sebagai berikut:

$$C_j = I + \frac{\gamma^2}{\sigma^2} (\bar{V}_{RF}^*)^H F_1 \bar{V}_{RF}^j \tag{2.28}$$

$$\eta_{ij} = \textstyle\sum_{l \neq i} G_j(i,l) V_{RF}(l,j) \tag{2.29}$$

$$\zeta_{ij} = G_j(i,i) + 2\, Re\left\{ \textstyle\sum_{m \neq i, n \neq i} V_{RF}^*(m,j) G_j(m,n) V_{RF}(n,j) \right\} \tag{2.30}$$

$$G_j = \frac{\gamma^2}{\sigma^2} F_1 + \frac{\gamma^4}{\sigma^4} F_1 \bar{V}_{RF}^j C_j^{-1} \left(\bar{V}_{RF}^j \right)^H F_1 \tag{2.31}$$

Dimana $\gamma = \sqrt{P/(N\, N^{RF})}$. Dengan $\bar{V}_{RF}^j$ adalah submatriks dari V_{RF} dengan kolom j^{th} yang dihilangkan. Jika mengasumsikan semua elemen dari RF *precoder* adalah tetap kecuali untuk $V_{RF}(i,j)$ dan C_j, η_{ij}, ζ_{ij} independen untuk $V_{RF}(i,j)$, maka nilai optimal untuk elemen matriks RF *precoder* pada baris ke-*i* dan kolom ke-*j* adalah:

$$V_{RF}(i,j) = \begin{cases} 1, & jika\ \eta_{ij} = 0 \\ \frac{\eta_{ij}}{|\eta_{ij}|}, & lainnya \end{cases} \tag{2.32}$$

Untuk mendapatkan *hybrid combiner* yang memaksimalkan efisiensi spektral secara keseluruhan dengan menggunakan jumlah RF *chain* yang sama dengan jumlah aliran data ($N^{RF} = N_s$), desain W_{RF} dan W_D dapat dipisahkan terlebih dahulu dengan merancang RF *combiner* dengan asumsi *digital compiner* optimal untuk selanjutnya menemukan *digital combiner* untuk RF *combiner* tersebut. Maka, permasalah untuk menemukan RF *combiner* dapat ditulis sebagai berikut:

$$\max_{W_{RF}} \quad log_2 \left| I + \frac{1}{\sigma^2} (W_{RF}^H W_{RF})^{-1} W_{RF}^H F_2 W_{RF} \right| \tag{2.33a}$$

$$|\mathrm{W_{RF}}(i,j)|^2 = 1, \ \forall i,j \tag{2.33b}$$

Dimana $F_2 = HV_tV_t^H H^H$, dibutuhkan untuk mencari W_{RF} yang memenuhi $W_{RF}^H W_{RF} \approx IM$ untuk jumlah M yang besar. Permasalahan W_{RF} pada persamaan 3.16, dapat diselesaikan dengan menggunakan algoritma yang sama untuk mencari V_{RF} dengan mensubstitusi F_1 dan γ^2 dengan F_2 dan $\frac{1}{M}$. Jika semua *beamformers* sudah mencapai nilai yang tetap, maka *digital combiner* yang optimal dapat dirumuskan dengan persamaan 3.17.

$$W_D = J^{-1} W_{RF}^H H V_t \tag{2.34}$$

Dimana $J = W_{RF}^H H V_t V_t^H H^H W_{RF} + \sigma^2 W_{RF}^H W_{RF}$.

2.5.3 Point-to-Point LSA-MIMO berbasis FDB

Pada acuan referensi [18], sistem *point-to-point* LSA-MIMO berbasis *full-digital beamforming* dapat dicapai dengan menggunakan algoritma yang sama untuk memodelkan sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming*. Namun pada pemodelan FDB, jumlah RF *chain* yang digunakan adalah sebanyak dua kali jumlah RF *chain* yang digunakan pada sistem *point-to-point* LSA-MIMO berbasis HB.

2.6 Estimasi BER

Bit Error Rate (BER) atau teknik perhitungan jumlah kesalahan yang digunakan adalah dengan menggunakan rata-rata sampel. Jika dalam sistem *b* bit yang tidak sesuai dengan bit dikirim diawal (error) dan terdapat sebanyak *B* bit informasi yang dikirimkan, maka BER dapat

dihitung dengan menggunakan persamaan berikut:

$$P_o = \frac{jumlah\ bit\ error\ (b)}{total\ bit\ yang\ dikirimkan\ (B)} \tag{2.35}$$

2.7 Efisiensi Spektral

Efisiensi spektral merupakan tingkat kecepatan atau laju transmisi informasi yang dapat dikirimkan melalui ukuran *bandwidth* tertentu dalam suatu sistem. Pada sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming*, didapatkan efisiensi spektral sebagai berikut:

$$R = \log 2 \left| \mathbf{I}_M \frac{1}{\sigma^2} \mathrm{W_t} (\mathrm{W_t^H W_t})^{-1} \mathrm{W_t^H H V_t V_t^H H^H} \right| \tag{2.36}$$

Dimana $\mathbf{I}_M$ merupakan matriks identitas berukuran M, $\mathrm{W_t}$ merupakan hasil perkalian dari RF *combiner,* $\mathrm{W_{RF}}$ dan *digital combiner,* $\mathrm{W_D}$. $\mathrm{V_t}$ merupakan hasil perkalian dari RF *precoder,* $\mathrm{V_{RF}}$ dan *digital precoder,* $\mathrm{V_D}$. Serta H merupakan matriks kanal dan σ^2 merupakan varians sistem.

2.8 Pola Radiasi

Pola radiasi yang disimulasikan menggunakan rumus *array factor* pada persamaan 2.17. Dimana pola tersebut didapatkan dari antena jamak sejumlah N dan pola yang didapatkan merupakan pola radiasi pada medan jauh antena. Parameter A_n menggunakan magnitud dari matriks V_t dan α yang merupakan fase dari matriks V_t.

$$AF(\psi) = \sum_{n=0}^{N-1} A_n e^{in\beta d \cos\theta + \alpha} \tag{2.37}$$

$$f(\psi) = \frac{AF(\psi)}{\max\left[AF(\psi)\right]} \tag{2.38}$$

$F(\psi)$ merupakan normalisasi dari $AF(\psi)$. Dengan tetapan fase, $\beta = \frac{2\pi}{\lambda}$, θ merupakan sudut yang dibentuk antara elemen antena pada BS dengan elemen antena pada UE dan d adalah jarak antara elemen antena [20].

[Halaman Ini Sengaja Dikosongkan]

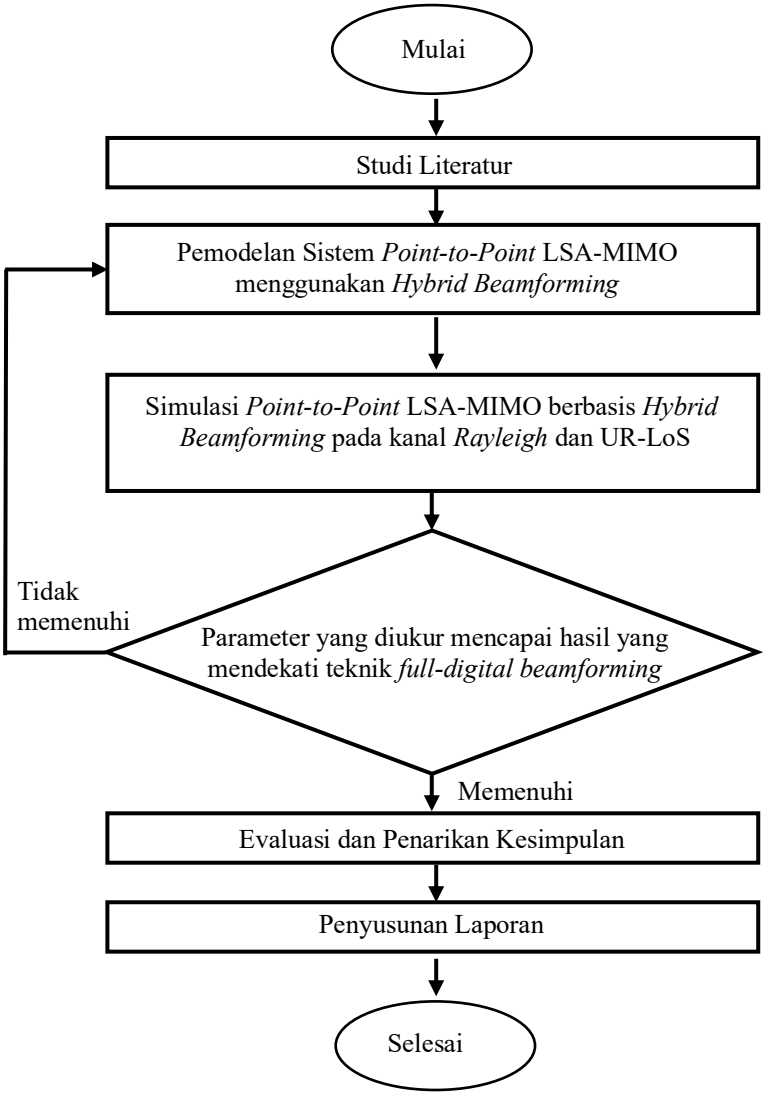
BAB 3
METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan mengenai tahapan-tahapan yang dilakukan dalam proses pembuatan tugas akhir ini. Tugas akhir ini membahas mengenai perbandingan kinerja sistem *point-to-point* LSA-MIMO pada kanal *independent Rayleigh fading* dan *uniformly Random Line-of-Sight* (UR-LoS) yang ditunjukan dengan kurva *Signal-to-Noise Ratio* (SNR) terhadap efisiensi spektral. SNR merupakan perbandingan antara sinyal dengan *noise*, sedangkan spektral efisiensi adalah kemampuan sistem mengalokasikan spektrum frekuensi secara efisien dan semaksimal mungkin dalam frekuensi yang terbatas. Dalam pemodelan sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming*, simulasi dijalankan menggunakan *software Matlab* R2020a (9.8.0) versi 20.0.

3.1 Parameter Sistem

Simulasi perancangan sistem *point-to-point* LSA-MIMO berbasis *Hybrid Beamforming* seperti pemodelan yang terdapat pada Gambar 3.1 diperlukan sebelum melakukan analisa tugas akhir. Pada simulasi ini, parameter dan asumsi yang digunakan dalam sistem adalah:

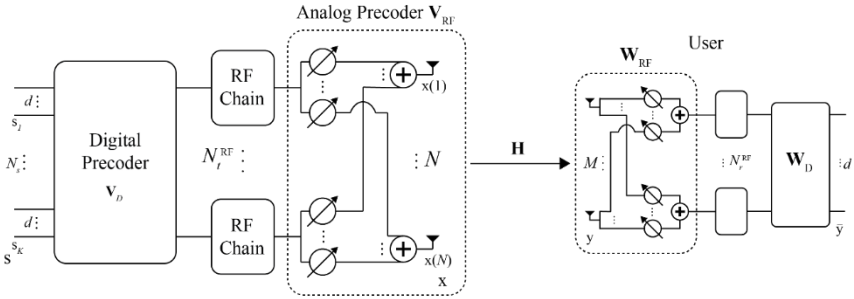
a. Input data merupakan data biner yang menggunakan modulasi BPSK.
b. Sistem yang digunakan adalah *point-to-point* LSA-MIMO.
c. Konfigurasi sistem yang divariasikan untuk melihat pengaruhnya terhadap kinerja sistem adalah jumlah antena *base station* (BS), *user equipment* (UE), dan RF *chain.*
d. Kanal yang digunakan adalah *independent Rayleigh fading* (IRF) dan *uniformly Random Line-of-Sight* (URLoS).
e. Performansi sistem yang akan dibahas adalah perbandingan efisiensi spektral (*spectral efficiency)* terhadap fungsi SNR (*Signal-to-Noise Ratio).*
f. Metode *hybrid beamforming* (HB) dimodelkan dengan menggunakan jumlah RF *chain* sebanyak setengah dari jumlah RF *chain* yang digunakan untuk *full-digital beamforming* (FDB).



Gambar 3.1 Diagram alur simulasi Tugas Akhir

3.2 Pemodelan Sistem

Sistem yang akan disimulasikan ditunjukkan pada Gambar 3.2, dimana selanjutnya akan dilakukan pengujian dua jenis kanal yang berbeda, *independent Rayleigh fading* dan UR-LoS, terhadap sistem tersebut. Terdapat beberapa bagian dalam sistem diagram blok diatas, yaitu *digital precoder* (V_d), *analog precoder* (V_{RF}), kanal (H), RF *combiner* (W_{RF}), *digital combiner* (W_D), dan RF *chain* (N^{RF}).

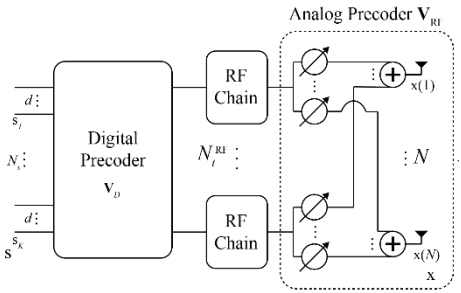


Gambar 3.2 Diagram blok point-to-point LSA-MIMO berbasis HB

Dalam sistem *point-to-point* LSA-MIMO berbasis *Hybrid Beamforming*, terdapat *N* antena pemancar dan satu *user* (*K*) dengan *M* antena penerima. Pada sisi pemancar, terdapat sejumlah *d* aliran data yang dikirim menuju *user* yang telah ditentukan. Aliran data tersebut diterima oleh V_d untuk dilakukan pengolahan data secara digital, setelah itu dilakukan penggeseran frekuensi menuju frekuensi pembawa dengan melewatkannya pada RF *chain* pemancar (N_t^{RF}). Untuk mencapai performansi yang hampir menyerupai *full-digital beamforming,* maka diasumsikan jumlah N^{RF} yang digunakan pada kedua sisi penerima dan pemancar sama dengan jumlah aliran data (*d*) yang digunakan. Sementara itu, performa *full-digital beamforming* sendiri dapat direalisasikan dengan menggunakan N_t^{RF} sejumlah dua kali dari jumlah aliran data yang digunakan[18].

3.2.1 Bagian *Transmitter*

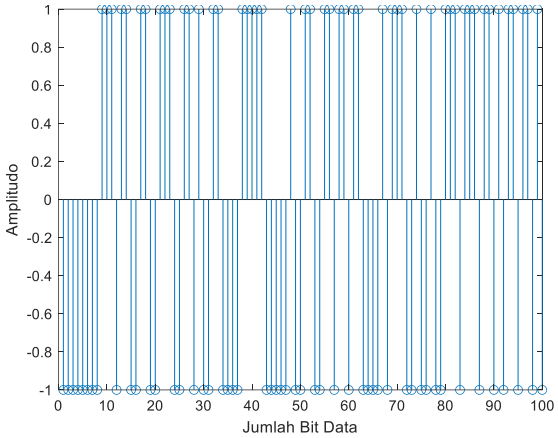
Proses yang terjadi pada bagian *transmitter* terdiri dari beberapa tahap seperti pembangkitan bit data, pengolahan sinyal dengan *digital precoder* (V_d), penguatan sinyal dengan RF *chain* (N^{RF}), pengolahan sinyal kembali dengan *analog precoder* (V_{RF}) yang kemudian dipancarkan antena pemancar. Bit data input dibangkitkan untuk kemudian akan melalui modulator untuk melalui proses modulasi. Modulasi yang digunakan adalah BPSK dengan bit yang dibangkitkan adalah sejumlah 6000 bit data.



Gambar 3.3 Blok diagram pemancar *Point-to-Point* LSA-MIMO

3.2.1.1 *Pembangkitan bit input dengan BPSK*

Sinyal input dibangkitkan secara acak, pertama dengan *command rand* pada MATLAB untuk membangkitkan angka secara acak dengan interval dari 0 sampai 1. Selanjutnya dengan menggunakan fungsi *logical* angka yang dibangkitkan lebih besar dari 0.5, akan didapatkan data yang terdistribusi acak dengan nilai satu dan nol. Data tersebut dikalikan dua dan dikurangi satu untuk mendapatkan modulasi BPSK yang terdistribusi dari -1 sampai 1. Pada Tugas Akhir ini, jumlah *bit* informasi yang digunakan adalah sebanyak 6000 bit informasi dengan 6 aliran data (*data stream*), dimana masing-masing aliran data membangkitkan 1000 bit informasi. Dengan mengambil 100 *bit* sampel dari 6000 *bit* maka didapatkan Gambar 3.4.



Gambar 3.4 Sampel stem Bit informasi pada input pemancar

3.2.1.2 *Pemodelan Digital Precoder*

Matriks dari elemen *digital preocoder* dimodelkan dengan melakukan perhitungan pada persamaan 2.19, dimana sebelumnya perlu didapatkan terlebih dahulu nilai elemen dari matriks RF *precoder* yang sudah mencapai konvergen.

3.2.1.3 *Pemodelan RF Precoder*

Untuk mendapatkan algoritma RF *precoder* yang memenuhi persamaan 2.18b, seperti pada awalnya dengan menetukan matriks V_{RF} yang memiliki ukuran $V_{RF}^{(0)} = 1_{N\,x\,N^{RF}}$, untuk kemudian masing-masing elemen diperbaharui dengan merujuk persamaan 2.26 sampai algoritma dari RF *precoder* mencapai konvergen dengan persamaan 2.18b.

3.2.2 Pemodelan Kanal

Pada Tugas Akhir ini, perbandingan performansi dilihat dalam dua kanal yang berbeda, yaitu *independent Rayleigh fading (*IRF) dan *uniformly random Line-of-Sight* (URLoS). Perbedaan dari dua kanal tersebut terdapat pada jalur transmisi yang dilewati sinyal informasi dari sisi pemancar sampai dengan sisi penerima. Pada Tugas Akhir kali ini digunakan nilai SNR sebesar -10 hingga 30 dB.

3.2.2.1 *Kanal Uniformly Random Line-of-Sight*

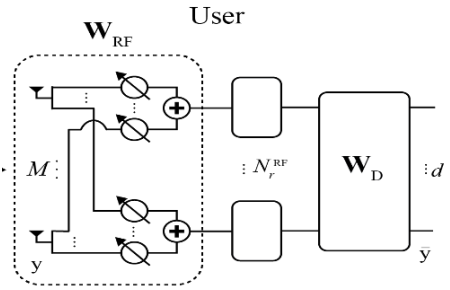
Pada kanal URLoS, pemancar dapat melihat secara langsung penerima dalam suatu garis lurus, atau dengan kata lain, tidak terdapat objek yang menghalangi pemancar dan penerima yang dapat menyebabkan pantulan. Pada Tugas Akhir, ditentukan UE memiliki sudut sebesar 6° terhadap BS untuk sistem dengan satu UE. Sedangkan untuk sistem dengan dua UE, UE 1 memiliki sudut sebesar 6° dan UE 2 memiliki sudut sebesar 60° terhadap BS.

3.2.2.2 *Kanal Independet Rayleigh Fading*

Kanal *Independent Rayleigh Fading* (IRF) merupakan kanal yang terdistribusi secara *Rayleigh*, dimana tidak terdapat jalur langsung antara pemancar dan penerima. Kanal IRF mempunyai elemen acak independent dengan rerataan nol dan distribusi bilangan kompleks $(0,\beta_k)$ dimana β_k mempunyai nilai 1 [16]. Untuk kanal IRF, sudut antara UE dan BS dibangkitkan secara acak untuk setiap pembangkitan kanal IRF.

3.2.3 Bagian Receiver

Pada sisi penerima, terdapat dua komponen utama dalam sistem *point-to-point* LSA-MIMO. Pertama adalah RF *combiner*, W_{RF}, dan yang kedua adalah *digital combiner*, W_D. Pemodelan W_{RF} dapat dilakukan dengan menggunakan algoritma yang sama dengan algoritma untuk membangkitkan V_{RF}, namun dengan mengganti nilai F_1 dan γ^2 dengan F_2 dan $\frac{1}{M}$. Sedangkan, untuk membangkitkan nilai matriks W_D dapat dilakukan dengan menghitung persamaan 2.34.



Gambar 3.5 Blok diagram Point-to-Point LSA-MIMO sisi penerima

3.2.3.1 Demapping

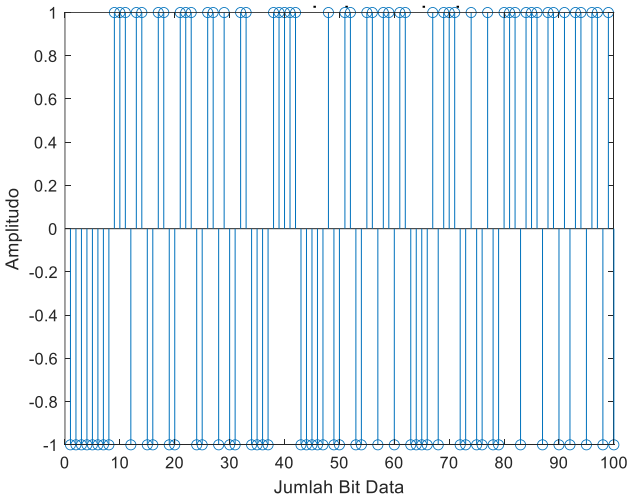
Proses *demapping* atau pengembalian bentuk *bit* informasi ke bentuk BPSK seperti yang diinputkan pada pemancar dilakukan dengan persamaan 3.18 sampai 3.20 dibawah ini.

$$sinyal\ ouput\ pemancar\ = V_t \times s \tag{3.18}$$

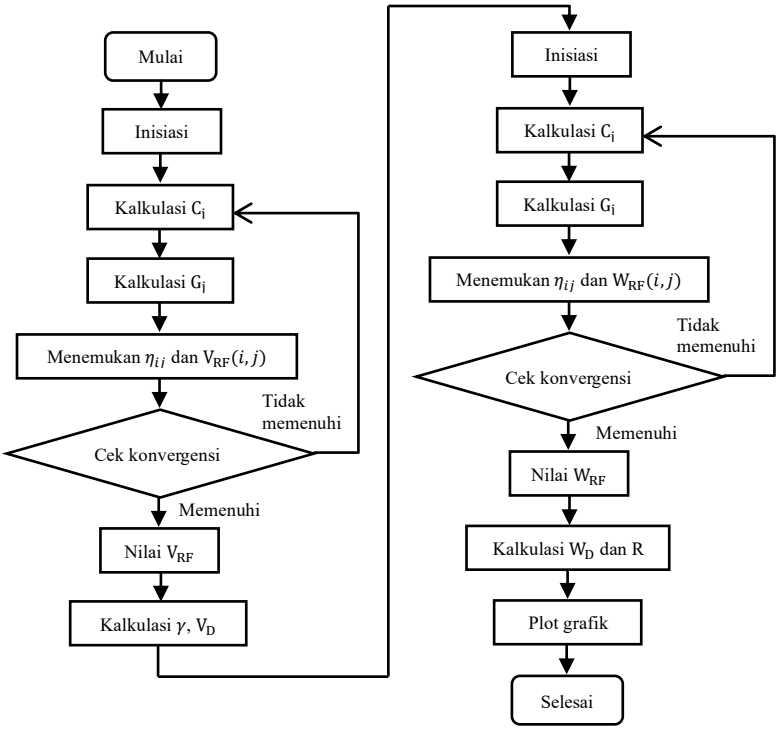
$$sinyal\ input\ penerima\ = H \times sinyal\ output\ pemancar \tag{3.19}$$

$$sinyal\ output\ penerima\ = W_t \times sinyal\ input\ penerima \tag{3.20}$$

Dimana *s* adalah matriks jumlah aliran data yang digunakan pada sistem berukuran $N_s \times 1$. Sinyal output penerima dalam persamaan 3.20 digunakan untuk mengetahui nilai kesalahan bit informasi atau BER (*bit error rate*) yang diterima oleh penerima, dibandingkan dengan bit sinyal informasi yang dibangkitkan pada input penerima. Gambar 3.8 menampilkan sampel 100 *bit* pertama dari sinyal output penerima.



Gambar 3.6 Sampel stem bit informasi pada keluaran penerima



Gambar 3.7 Alur diagram metodologi penelitian Tugas Akhir

Secara singkat, alur penelitian pada Tugas Akhir ini dapat dilihat pada Gambar 3.7. Simulasi dimulai dengan mencari nilai dari matriks *analog precoder*, V_{RF}. Hasil dari matriks V_{RF} digunakan untuk mencari nilai dari matriks *digital precoder*, V_D. Kemudian nilai matriks V_{RF} dan V_D digunakan untuk mencari nilai dari matriks RF *combiner*, W_{RF}. Nilai dari ketiga matriks tersebut digunakan untuk mencapatkan nilai dari matriks *digital combiner*, W_D. Setelah semua nilai matriks V_{RF}, V_D, W_{RF} dan W_D didapatkan, maka efisiensi spektral dapat dihitung dengan persamaan 2.36.

3.3 Simulasi Sistem *Point-to-Point* LSA-MIMO

Variasi konfigurasi sistem dilakukan pada simulasi sistem *point-to-point* LSA-MIMO pada kanal IRF dan URLoS untuk melihat pengaruh pada kinerja efisiensi spektral sistem. Beberapa variasi konfigurasi pada

sistem yang disimulasikan adalah sebagai berikut:

a. Simulasi sistem *point-to-point* LSA-MIMO berbasis FDB dan HB pada kanal IRF dan URLoS.
b. Simulasi sistem *point-to-point* LSA-MIMO berbasis HB dengan variasi jumlah Tx dan Rx pada kanal IRF dan URLoS.
c. Simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal IRF dan URLoS.
d. Simulasi sistem *point-to-point* LSA-MIMO dengan metode FDB dan HB untuk 2 UE pada kanal IRF dan URLoS.
e. Simulasi pola radiasi BS dan UE pada sistem *point-to-point* LSA-MIMO.

Parameter yang turut disumulasikan untuk mendukung hasil pengujian sistem *point-to-point* LSA-MIMO adalah BER, SE, dan pola radiasi.

3.3.1 Simulasi Analisa BER

Sebelum menganalisa efisiensi spektral dari sistem *point-to-point* LSA-MIMO dengan *hybrid beamforming*, lebih dulu dilakukan Analisa pada BER sistem untuk memeriksa apakah sistem dapat mengirimkan *bit* sinyal informasi dari input pemancar sampai dengan output pemancar dengan baik. Semakin tinggi nilai BER maka semakin butuk kinerja dari sistem *point-to-point* LSA-MIMO tersebut, dan sebaliknya jika nilai BER semakin kecil maka kinerja sistem lebih baik. Persamaan 2.15 digunakan untuk menganalisa kinerja BER dari sistem.

3.3.2 Simulasi Analisa Efisiensi Spektral

Pada Tugas Akhir ini, dilakukan iterasi pada sistem sebanyak 100 kali. Rata-rata dari semua hasil iterasi tersebut digunakan untuk mebuat *plot* kurva efisiensi spektral terhadap SNR. Analisa efisiensi spektral sistem *point-to-point* LSA-MIMO dengan *hybrid beamforming* dilakukan dengan melakukan perhitungan pada persamaan 2.16, kemudian dengan melihat bentuk kurva efisiensi spektral terhadap SNR.

3.3.3 Simulasi Pola Radiasi BS dan UE

Untuk melihat bagaimana teknik *beamforming* mempengaruhi pola radiasi pada sisi BS dan UE sistem, maka disimulasikan pola radiasi dengan menggunakan persamaan *array factor* yang tertera pada persamaan 2.17.

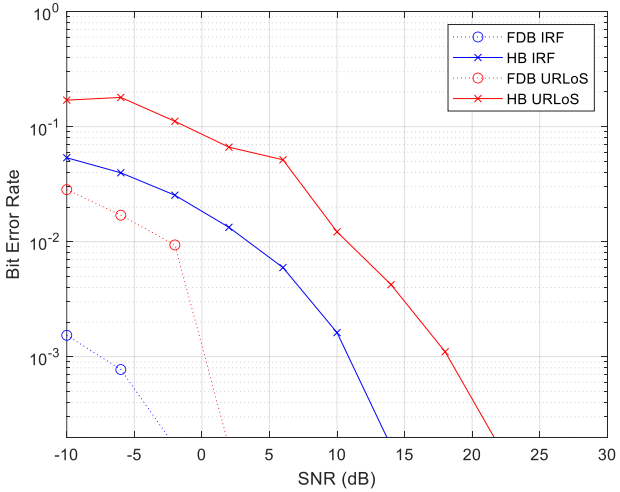
[Halaman Ini Sengaja Dikosongkan]

BAB 4
PENGAMBILAN DAN ANALISA DATA

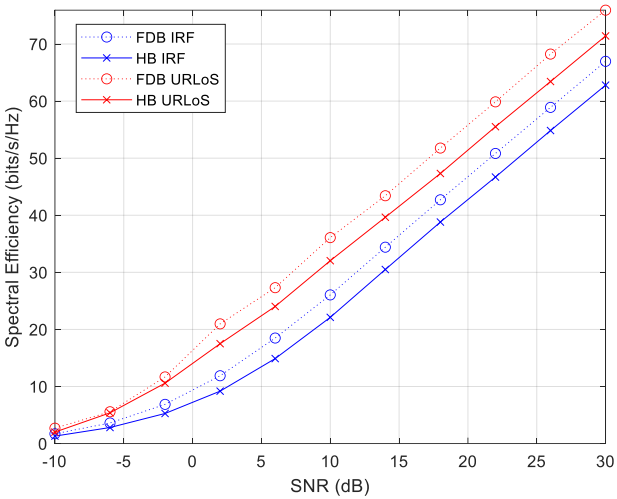
Pada bab pengambilan dan Analisa data, akan ditampilkan hasil dari simulasi sistem *point-to-point* LSA-MIMO dengan metode *hybrid beamforming* (HB) dan *full-digital beamforming* (FDB) pada dua kanal yang berbeda, yaitu *uniformly random line-of-sight* (URLoS) dan *independent Rayleigh fading* (IRF). Hasil dari simulasi akan dianalisa untuk mengetahui kinerja sistem *point-to-point* LSA-MIMO terhadap kedua kanal tersebut, serta perbandingan kinerja untuk berbagai konfigurasi yang dilakukan pada sistem *point-to-point* LSA-MIMO. Untuk masing-masing simulasi, dilakukan 100 kali iterasi sistem.

4.1 Hasil simulasi sistem *point-to-point* LSA-MIMO berbasis FDB dan HB pada kanal URLoS dan IRF

Pada sub bab ini, disimulasikan sistem *point-to-point* LSA-MIMO dengan perbandingan metode *full-digital beamforming* (FDB) dan *hybrid beamforming* (HB) pada masing-masing kanal IRF dan URLoS untuk menunjukkan kinerja metode *hybrid beamforming* yang dapat menyerupai kinerja metode *full-digital beamforming.*



Gambar 4.1 BER *point-to-point* LSA-MIMO berbasis FDB dan HB pada kanal IRF dan URLoS



Gambar 4.2 Grafik perbandingan SE sistem *point-to-point* LSA-MIMO berbasis FDB dan HB pada kanal IRF dan URLoS

Berdasarkan hasil simulasi terhadap kanal IRF dan URLoS pada sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming*, didapatkan perbandingan grafik BER yang ditunjukkan pada Gambar 4.1. Hasil yang didapatkan untuk sistem *point-to-point* LSA-MIMO berbasis HB pada kanal IRF, untuk pengamatan nilai BER 10^{-3}, kanal IRF memiliki kinerja SNR 10 dB. Sedangkan untuk kanal URLoS, memiliki kinerja SNR 15 dB pada nilai BER yang sama. Sementara sistem *point-to-point* LSA-MIMO berbasis FDB, untuk mencapai kinerja BER 10^{-3}, kanal IRF memiliki kinerja SNR -7 dB dan pada kanal URLoS memiliki kinerja SNR sebesar 0 dB. Hal ini menujukkan kinerja BER sistem *point-to-point* LSA-MIMO berbasis HB untuk kanal IRF lebih baik ± 5 dB dibandingkan dengan kanal URLoS pada nilai SNR yang sama, dan pada sistem berbasis FDB, kanal IRF memiliki kinerja SNR ± 7 dB lebih baik dibandingkan dengan kanal URLoS.

Setelah didapatkan pengamatan terhadap nilai BER sistem *point-to-point* LSA-MIMO berbasis HB, kemudian dilakukan pengujian dan pengamatan terhadap nilai *spectral efficiency* (SE) pada kanal IRF dan URLoS. Perbandingan nilai SE dapat dilihat dengan grafik pada Gambar 4.2 dan dilihat secara nilai pada Tabel 4.1. Pada Tabel 4.1, terlihat bahwa Ketika kedua kanal memiliki nilai SNR 10 dB, kanal IRF mempunyai SE

sebesar 22,0621 bits/s/Hz sementara kanal URLoS mempunyai SE sebesar 31,9902 bits/s/Hz pada nilai SNR yang sama. Hal ini menujukkan bahwa kanal URLoS memiliki SE ±9,9281 bits/s/Hz lebih baik dibandingkan dengan kanal IRF pada SNR yang sama, atau terjadi peningkatan SE sebesar 45% pada kanal URLoS dibandingkan dengan kanal IRF. Perbedaan kinerja sebesar 45% antara dua kanal tersebut stabil pada nilai SNR 10 dB keatas.

Untuk hasil simulasi terhadap SE dari sistem *point-to-point* LSA-MIMO pada kanal IRF dengan metode FDB dan metode HB dapat dilihat pada Gambar 4.2 untuk grafik dan nilai terhadap SNR pada Tabel 4.1. Pada Gambar 4.2, terlihat bahwa pada SNR 10 dB, kanal IRF dengan metode FDB mempunyai SE sebesar 26,04 bits/s/Hz, sedangkan pada metode HB, SE yang didapat adalah 22,11 bits/s/Hz. Terdapat peningkatan SE sebesar 3,93 bits/s/Hz pada metode FDB jika dibandingkan dengan metode HB, atau meningkat sebesar 17%. Jika dilihat pada nilai SE yang sama, yaitu 10 bits/s/Hz, metode FDB memiliki kinerja SNR 1 dB dan metode HB memiliki kinerja SNR 3 dB. Perbedaan SNR sebanyak 2 dB tersebut cenderung stabil pada nilai SNR diatas 0 dB. Sehingga, metode HB memiliki kinerja SE yang hampir menyerupai metode FDB dengan jumlah RF *chain* setengah dari jumlah RF *chain* yang digunakan metode FDB pada kanal IRF.

Simulasi untuk mengetahui kinerja SE pada sistem *point-to-point* LSA-MIMO berbasis FDB dan HB pada kanal URLoS terlihat pada Gambar 4.1 dan Tabel 4.2, dimana saat SNR 10 dB, metode FDB mempunyai kinerja SE 36,08 bits/s/Hz dan metode HB mempunyai kinerja SE 32,04 bits/s/Hz. Oleh sebab itu, pada kanal URLoS, kinerja SE metode HB dapat menyerupai kinerja SE metode FDB dengan selisih 4,04 bits/s/Hz. Terjadi peningkatan sebesar 12,61% untuk sistem FDB. Peningkatan tersebut cenderung stabil untuk kedua sistem pada nilai SNR 0 dB keatas.

Tabel 4.1 SE pada sistem *point-to-point* LSA-MIMO berbasis FDB dan HB dengan kanal IRF dan URLoS.

SNR (dB)	**Efisiensi Spektral (bits/s/Hz)**			
	Kanal			
	IRF		**URLoS**	
	FDB	**HB**	**FDB**	**HB**
-10	1,7196	1,2938	2,7007	1,9924
-6	3,6082	2,8645	5,5755	5,3499

-2	6,8703	5,2884	11,6763	10,5101
2	11,8827	9,2112	20,9714	17,5621
6	18,4890	14,8306	27,3101	24,1721
10	26,0447	22,1121	36,0804	32,0402
14	34,4046	30,3088	43,4185	39,5479
18	42,7105	38,7671	51,7709	47,4181
22	50,8285	46,8429	59,8673	55,4183
26	58,8966	54,6974	68,2327	63,7085
30	66,9638	62,8403	75,9277	71,5076

4.2 Hasil simulasi sistem *point-to-point* LSA-MIMO berbasis HB dengan variasi jumlah Tx dan Rx pada kanal IRF

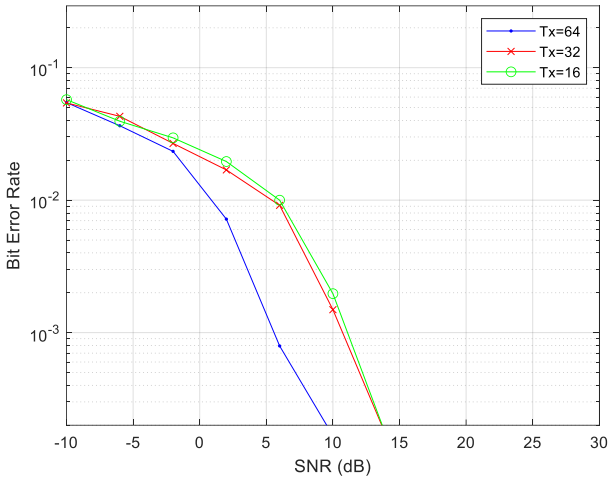
Simulasi kali ini memvariasikan jumlah antena pemancar (Tx) dan antena penerima (Rx) sistem *point-to-point* LSA-MIMO pada kanal IRF untuk melihat pengaruh jumlah antena pemancar dan penerima terhadap kinerja BER maupun SE. Jumlah antena Tx divariasikan dengan tiga variasi jumlah, yaitu Tx=64, Tx=32, dan Tx=16. Analisa BER dan SE akan dilakukan untuk mengamati pengaruh dari perubahan jumlah antena pada pemancar.

4.2.1 Hasil simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal IRF

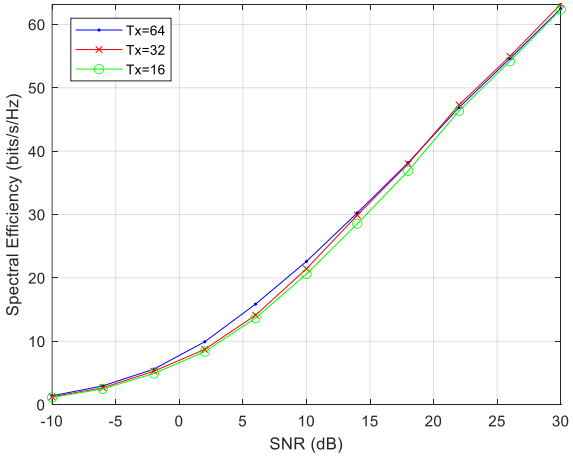
Berdasarkan hasil simulasi pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena pemancar (Tx) pada kanal IRF, didapatkan BER hasil simulasi seperti pada Gambar 4.3. Pada hasil simulasi, terlihat perbedaan yang cukup signifikan antara jumlah Tx=64 dengan jumlah Tx=32 dan Tx=16 saat nilai SNR mendekati 10 dB. Dengan SNR yang dibutuhkan untuk mendapatkan BER 10^{-3} pada Tx=64 adalah 5 dB, Tx=32 membutuhkan SNR 11 dB, dan Tx=16 membutuhkan SNR sebesar 11,5 dB. Perbedaan kinerja BER antara jumlah Tx=64 dengan jumlah Tx=32 dan Tx=16 cukup signifikan terjadi pada SNR -2 dB.

Seperti yang terlihat pada grafik 4.4 dan Tabel 4.2, kinerja SE untuk masing-masing Tx dimana Tx=64, Tx=32, dan Tx=16 memiliki kinerja SE yang menyerupai satu sama lain. Seperti yang terlihat pada Gambar 4.8, saat SNR memiliki nilai 10 dB, untuk Tx=64 memiliki kinerja SE sebesar 22,6 bits/s/Hz, untuk Tx=32 memiliki kinerja SE sebesar 21,43 bits/s/Hz, untuk Tx=16 memiliki kinerja SE sebesar 20,61

bits/s/Hz.



Gambar 4.3 Grafik BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal IRF



Gambar 4.4 Grafik SE sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal IRF

Membandingkan dengan kinerja metode FDB kanal IRF untuk mencapai kinerja SE 18,49 bits/s/Hz, sistem dengan Tx=64 membutuhkan nilai SNR sebesar 8 dB dan memiliki nilai BER ±0.0003 pada SNR tersebut, sistem dengan Tx=32 membutuhkan nilai SNR sebesar 9 dB dan memiliki nilai BER 0.002 pada SNR tersebut, dan sistem dengan Tx=16 membutuhkan nilai SNR sebesar 9,5 dB dan memiliki BER 0,0025 pada SNR tersebut.

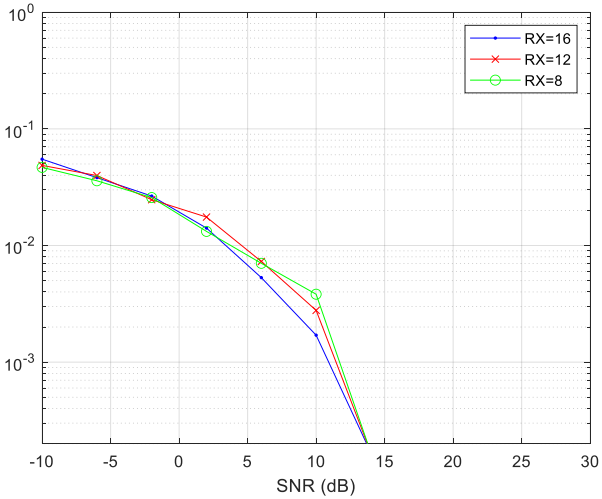
Tabel 4.2 SE pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena Tx pada kanal IRF

SNR (dB)	Efisiensi Spektral (bits/Hz) IRF		
	Jumlah antena pemancar (Tx)		
	64	32	16
-10	1,3913	1,2366	1,1332
-6	2,9667	2,6697	2,4894
-2	5,6029	5,2746	4,9133
2	9,9179	8,7187	8,3553
6	15,8659	14,1191	13,6664
10	22,5984	21,4307	20,6095
14	30,2631	29,8993	28,5487
18	38,1807	38,0138	36,8830
22	46,8860	47,3120	46,3690
26	54,6125	54,9697	54,1944
30	62,4961	63,1975	62,3511

4.2.2 Hasil simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal IRF

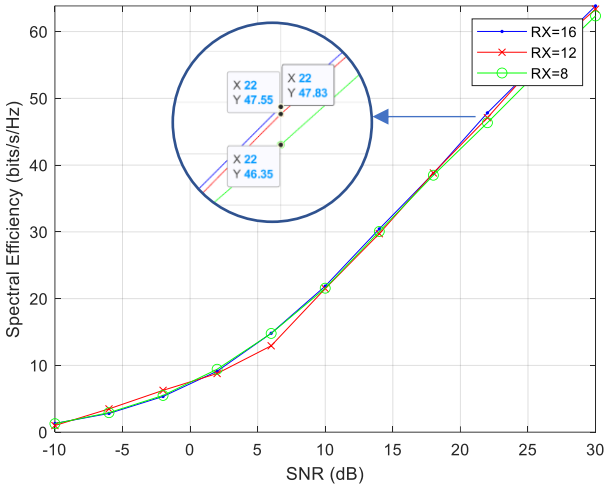
Pada Gambar 4.9, terlihat kinerja BER pada jumlah antena penerima (Rx) 16, 12, dan 8 tidak terdapat perbedaan yang cukup signifikan dan terlihat hampir menyerupai satu sama lain. Pada BER 10^{-3}, antena Rx 32 mempunyai SNR 11 dB, antena Rx 16 mempunyai SNR 12 dB, dan antena Rx 8 mempunyai kinerja SNR 12,5 dB.

Kinerja SE dengan variasi antena Rx dapat dilihat pada Gambar 4.10 dan Tabel 4.5, didapatkan kinerja SE yang sangat menyerupai untuk semua variasi jumlah Rx. Namun, pada nilai SNR mendekati 22 dB, jumlah Rx=16 memiliki kinerja SE 47,55 bits/s/Hz, sedikit diatas kinerja sistem dengan Rx=12 yang memiliki kinerja SE 47,83 bits/s/Hz dan Rx=8 yang memiliki kinerja SE 46,35 bits/s/Hz. Kinerja SE tersebut bertahan sampai SNR diatas 22 dB. Kinerja BER dan SE pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena penerima (Rx) pada kanal IRF cenderung stabil untuk rentang SNR -10 dB sampai 30 dB.



Gambar 4.5 Grafik BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal IRF

Membandingkan dengan kinerja metode FDB kanal IRF untuk mencapai kinerja SE 18,49 bits/s/Hz, Rx=16, Rx=12, dan Rx=8 membutuhkan nilai SNR yang hampir sama yaitu sekitar 8 dB, namun pada kinerja BER pada SNR 8 dB terjadi sedikit perbedaan pada ketiga variasi konfigurasi, pada Rx=16 didapatkan kinerja BER 0,002, pada Rx=12 didapatkan kinerja BER 0,0035 dan pada Rx=8 didapatkan kinerja BER 0,0045.



Gambar 4.6 Grafik SE sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal IRF

Tabel 4.3 SE pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal IRF

SNR (dB)	Efisiensi Spektral (bits/s/Hz) IRF		
	Jumlah antena penerima (Rx)		
	16	12	8
-10	1,3055	1,3278	1,2762
-6	2,8049	2,8517	2,9186
-2	5,3470	5,2766	5,4976
2	9,1316	8,9771	9,4231
6	14,8104	14,4158	14,7873
10	21,8725	21,6036	21,5628
14	30,4437	30,0584	30,0632
18	38,7350	38,3513	38,4830
22	47,8261	47,5486	46,3500
26	55,8789	55,5140	54,4913
30	63,8420	63,4420	62,3469

4.3 Hasil simulasi sistem *point-to-point* LSA-MIMO berbasis HB dengan variasi jumlah Tx dan Rx pada kanal URLoS

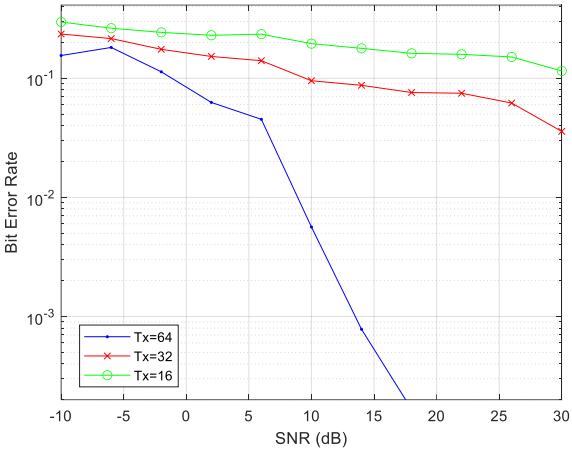
Sama seperti bab sebelumnya, namun pada bab ini sistem sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena pemancar (Tx) dan antena penerima (Rx) akan diuji coba pada kanal URLoS untuk mengamati dan menganalisa perubahan kinerja akibat variasi tersebut.

4.3.1 Hasil simulasi sistem *point-to-point* LSA-MIMO berbasis HB dengan variasi jumlah Tx pada kanal URLoS

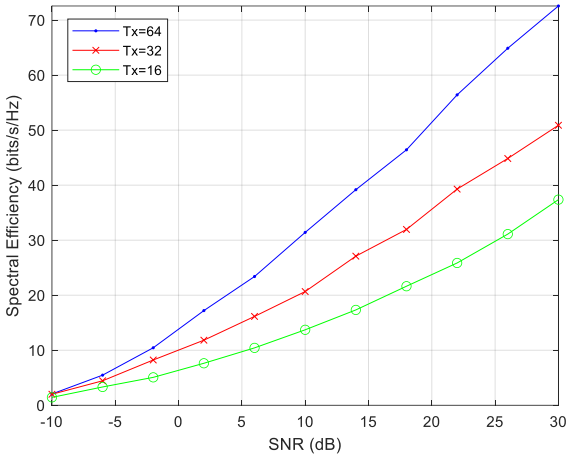
Analisa kinerja BER pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena pemancar (Tx) pada kanal URLoS dapat dilihat pada Gambar 4.7. Dimana terlihat perbedaan yang sangat signifikan untuk kinerja BER pada SNR 0 dB keatas dengan jumlah Tx=64 dengan jumlah Tx=32 dan Tx=16. Namun, perbedaan kinerja antara jumlah Tx=32 dan Tx=16 cenderung stabil pada nilai SNR 0 dB keatas. Untuk mencapai kinerja BER 10^{-1}, sistem dengan jumlah Tx=64 memiliki kinerja SNR 0,1 dB, sistem dengan jumlah Tx=32 memiliki kinerja SNR 10 dB, dan sistem dengan jumlah Tx=16 memiliki kinerja SNR 30 dB. Sehingga, sistem dengan BER paling baik didapatkan pada sistem dengan jumlah Tx=64 dan paling buruk pada sistem dengan jumlah Tx=16. Dapat disimpulkan bahwa kinerja sistem dengan Tx=64 jauh lebih baik dibandingkan dengan penggunaan jumlah Tx=32 dan Tx=16.

Analisa kinerja SE dilakukan dengan melihat grafik pada Gambar 4.8 dan nilai kinerja pada Tabel 4.4. Berdasarkan hasil simulasi terhadap sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal URLoS, dengan nilai SNR yang sama pada 10 dB, kinerja SE sistem dengan jumlah Tx=64 adalah sebesar 31,41 bits/s/Hz, sistem dengan jumlah Tx=32 sebesar 20,67 bits/s/Hz, dan sistem dengan jumlah Tx=16 sebesar 13,72 bits/s/Hz. Terdapat peningkatan sebesar 6,95 bits/s/Hz atau sebesar 50,65% pada jumlah Tx=32 dibandingkan jumlah Tx=16, peningkatan sebesar 10,74 bits/s/Hz atau sebesar 51,96% pada jumlah Tx=64 dibandingkan dengan jumlah Tx=32, dan peningkatan sebesar 128,94% pada jumlah Tx=64 dibandingkan dengan jumlah Tx=16. Semakin tinggi SNR, maka perbedaan yang terjadi diantara ketiga variasi jumlah Tx semakin lebar. Maka dapat disimpulkan bahwa jumlah Tx sangat berpengaruh terhadap kinerja SE sistem *point-to-point* LSA-

MIMO pada kanal URLoS.



Gambar 4.8 Grafik BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal URLoS



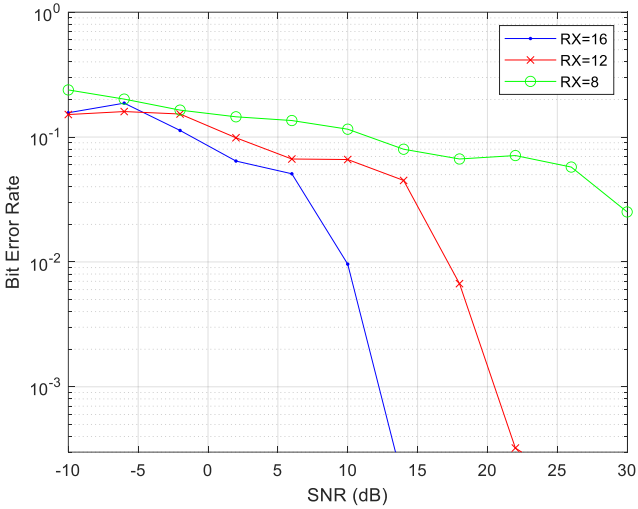
Gambar 4.7 Grafik SE sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal URLoS

Tabel 4.4 SE pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal URLoS

SNR (dB)	Efisiensi Spektral (bits/s/Hz) URLoS		
	Jumlah antena pemancar (Tx)		
	64	32	16
-10	2,0510	1,9990	1,4320
-6	5,4729	4,4464	3,3265
-2	10,4474	8,2287	5,0725
2	17,2043	11,8257	7,6315
6	23,3952	16,1479	10,4296
10	31,4128	20,6680	13,7154
14	39,1748	27,0996	17,3266
18	46,4260	31,9535	21,6336
22	56,4089	39,2871	25,8613
26	64,8797	44,8463	31,1039
30	72,5642	50,8851	37,3694

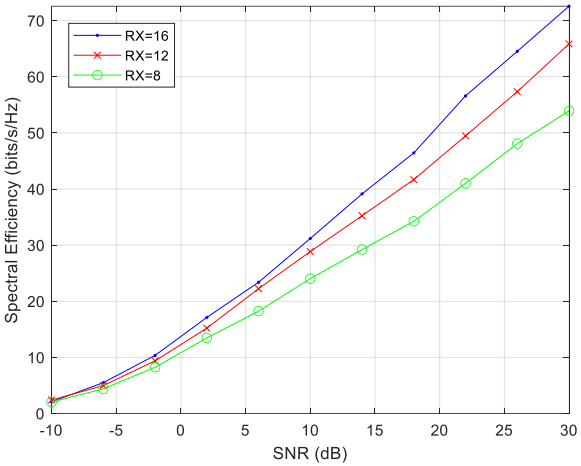
4.3.2 Hasil simulasi sistem *point-to-point* LSA-MIMO berbasis HB dengan variasi jumlah Rx pada kanal URLoS

Simulasi pada subbab ini akan memperlihatkan hasil simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena penerima (Rx) pada kanal URLoS. Analisa akan dilakukan dengan melihat Gambar dan Tabel dengan parameter pengamatan SNR terhadap SE dan SNR terhadap BER yang didapat dari hasil simulasi. Seperti yang terlihat pada Gambar 4.9, terdapat perbedaan yang cukup signifikan antara ketiga variasi jumlah antena Rx. Pada nilai BER 10^{-1}, sistem dengan jumlah Rx=16 memiliki kinerja SNR -2 dB, sistem dengan jumlah Rx=12 memiliki kinerja SNR 2 dB, dan sistem dengan jumlah Rx=8 memiliki kinerja SNR 11 dB. Dari hasil tersebut maka dapat disimpulkan bahwa jumlah Rx=16 lebih baik dibandingkan dengan Rx=12 dan Rx=8 untuk sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming* pada kanal URLoS.



Gambar 4.9 Grafik BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal URLoS

Kinerja SE pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx dapat dilihat pada Gambar 4.10 dan Tabel 4.5. Sesuai hasil yang terlihat pada Gambar dan grafik tersebut, sistem dengan jumlah Rx=16 memiliki kinerja SE yang paling baik dibandingkan dengan sistem dengan jumlah Rx=12 dan Rx=8. Pada SNR 10 dB, sistem dengan jumlah Rx=16 memiliki kinerja SE 31,64 bits/s/Hz, sistem dengan jumlah Rx=12 memiliki kinerja SE 28,86 bits/s/Hz, dan sistem dengan jumlah Rx=8 memiliki kinerja SE 24,02 bits/s/Hz. Pada Tabel 4.7 terlihat bahwa pada kinerja SE 40 bits/s/Hz, sistem dengan jumlah Rx=16 memiliki SNR 14 dB, sistem dengan jumlah Rx=12 memiliki SNR 17 dB, dan sistem dengan jumlah Rx=8 memiliki SNR 21 dB. Oleh karena itu, jumlah antena sangat berpengaruh terhadap kinerja SE pada sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming* pada kanal URLoS.



Gambar 4.10 Grafik SE sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal URLoS

Membandingkan dengan kinerja metode FDB kanal URLoS untuk mencapai kinerja SE 27,31 bits/s/Hz, sistem dengan Rx=16 membutuhkan nilai SNR sebesar 7 dB dan memiliki nilai BER ±0.03 pada SNR tersebut, sistem dengan Tx=32 membutuhkan nilai SNR sebesar 9 dB dan memiliki nilai BER 0,06 pada SNR tersebut, dan sistem dengan Tx=16 membutuhkan nilai SNR sebesar 13 dB dan memiliki BER 0,09 pada SNR tersebut.

Tabel 4.5 SE pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal URLoS

SNR (dB)	**Efisiensi Spektral (bits/s/Hz) URLoS**		
	Jumlah antena penerima (Rx)		
	16	**12**	**8**
-10	2,1120	2,3904	2,0367
-6	5,4493	4,9356	4,3869
-2	10,4182	9,4224	8,1722
2	17,1442	15,1384	13,4277
6	23,3982	22,0338	18,2551
10	31,6404	28,8909	24,0245

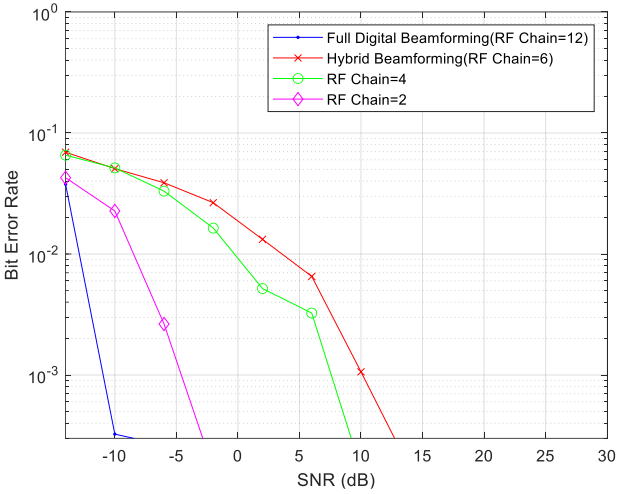
14	39,1039	35,3846	29,1951
18	46,2377	41,7226	34,3284
22	56,6798	49,3729	41,0266
26	64,5982	57,2538	48,0903
30	72,5532	65,9996	53,9689

4.4 Hasil simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal IRF dan URLOS

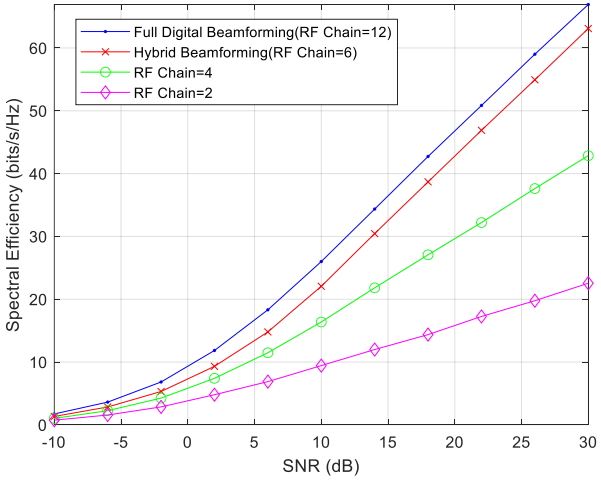
Pengamatan dan analisa pada subbab ini akan mengacu kepada pengaruh variasi jumlah RF *chain* yang digunakan oleh sistem *point-to-point* LSA-MIMO pada kanal IRF dan URLoS. Parameter yang ditinjau untuk dianalisa adalah BER dan SE yang didapatkan oleh hasil simulasi sistem *point-to-point* LSA-MIMO.

4.4.1 Hasil simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal IRF

Hasil simulasi kinerja BER terhadap variasi jumlah RF *chain* sistem *point-to-point* LSA-MIMO pada kanal IRF dapat dilihat pada Gambar 4.15. Pada nilai SNR lebih besar dari -10 dB, sistem dengan jumlah RF *chain* = 12 memiliki kinerja BER yang signifikan dibandingkan dengan sistem dengan jumlah RF *chain* lainnya dengan mengalami penurunan nilai BER yang lebih cepat, diikuti kinerja sistem dengan RF *chain* = 2, RF *chain* = 4, dan RF *chain* = 6. Urutan tersebut terjadi dimungkinkan terjadi akibat jumlah aliran data (Ns) yang digunakan berkurang pada sistem dengan RF *chain* = 4 dan RF *chain* = 2, dimana Ns yang digunakan mengikuti jumlah RF *chain* yang digunakan yaitu Ns = 4 untuk sistem dengan RF *chain* = 4 dan Ns = 2 untuk sistem dengan RF *chain* = 2. Dengan berkurangnya Ns, maka kemungkinan kesalahan pada pengiriman sinyal informasi juga berkurang.



Gambar 4.12 Grafik BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal IRF



Gambar 4.11 Grafik SE sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal IRF

Kinerja SE dengan variasi jumlah RF *chain* pada sistem *point-to-point* LSA-MIMO dapat dilihat pada Gambar 4.12 dan Tabel 4.6. Sesuai hasil pengamatan terhadap hasil simulasi, pergerakan grafik kinerja sistem dengan jumlah RF *chain* = 12 (*full-digital beamforming)* dan sistem dengan jumlah RF *chain* = 6 (*hybrid beamforming)* cenderung menyerupai dengan perbedaan kinerja SE sebesar 3,94 bits/s/Hz pada SNR 10 dB. Sementara sistem dengan jumlah RF *chain* = 4 memiliki kinerja SE 16,38 bits/s/Hz dan sistem dengan jumlah RF *chain* = 2 memiliki kinerja SE 9,436 bits/s/Hz pada SNR 10 dB, sehingga terjadi peningkatan sebesar 34,74 % pada sistem dengan RF *chain* = 6 terhadap sistem dengan RF *chain* = 4 dan peningkatan sebesar 133,89% pada sistem dengan dengan RF *chain* = 6 terhadap sistem dengan RF *chain* = 2. Sementara, terdapat perbedaan kinerja sebesar 17,85% pada RF *chain* = 12 dengan RF *chain* = 6. Perbedaan antara sistem dengan RF *chain* = 6 dengan sistem RF *chain* = 4 dan RF *chain* = 2 semakin melebar pada SNR diatas 0 dB. Maka dapat disimpulkan bahwa jumlah minimal RF *chain* yang digunakan pada metode *hybrid beamforming* untuk mencapai performa yang mendekati metode *full-digital beamforming* adalah setengah dari jumlah RF *chain* yang digunakan pada metode *full-digital beamforming.*

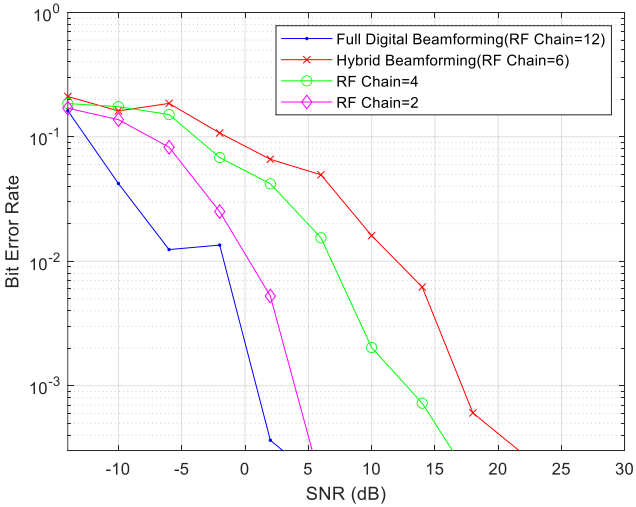
Membandingkan dengan kinerja metode FDB kanal IRF untuk mencapai kinerja SE 18,49 bits/s/Hz, sistem dengan RF *chain* = 4 membutuhkan nilai SNR sebesar 11 dB dan memiliki nilai BER 0.0002 pada SNR tersebut, sistem dengan RF *chain* = 2 membutuhkan nilai SNR sebesar 22 dB dan memiliki nilai BER 0.0002 pada SNR tersebut.

Tabel 4.6 SE pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal IRF

SNR (dB)	**Efisiensi Spektral (bits/s/Hz) IRF**			
	Jumlah RF *chain*			
	12	**6**	**4**	**2**
-10	1,7382	1,3439	1,0305	0,7442
-6	3,6256	2,8464	2,2627	1,5721
-2	6,8276	5,3176	4,2634	2,8613
2	11,8438	9,3178	7,4130	4,8015
6	18,3195	14,7996	11,4821	6,8855
10	26,0051	22,0671	16,3775	9,4358
14	34,3625	30,4522	21,8157	11,9936

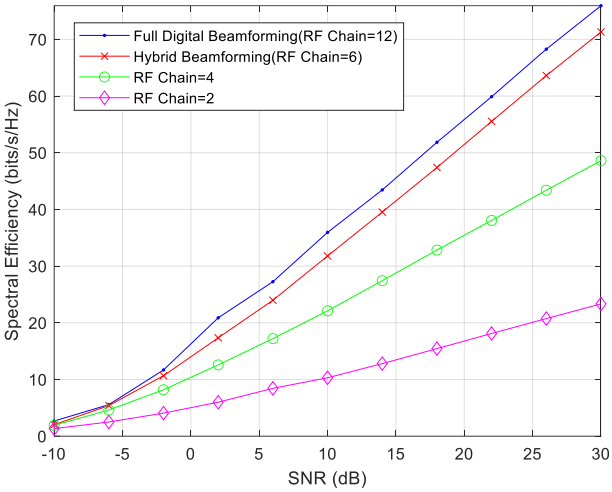
18	42,7330	38,6878	27,0763	14,3728
22	50,8500	46,8956	32,2074	17,2611
26	59,0101	54,9363	37,6185	19,7648
30	66,9199	63,0935	42,8611	22,5541

4.4.2 Hasil simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal URLoS



Gambar 4.13 Grafik BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal URLoS

Sama seperti simulasi pada subbab sebelumnya, simulasi subbab ini memvariasikan jumlah RF *chain* pada kanal URLoS dengan sistem *point-to-point* LSA-MIMO. Kinerja BER terhadap SNR pada sistem tersebut dapat dilihat pada Gambar 4.13. Pada SNR -30 dB, sistem dengan jumlah RF *chain* = 2 memiliki kinerja SNR paling baik dibandingkan sistem dengan jumlah RF *chain* lainnya. Namun, pada SNR -6 dB, kinerja BER sistem dengan RF *chain* = 12 meningkat lebih cepat dibandingkan dengan ketiga variasi lainnya. Sehingga dapat disimpulan bahwa sistem dengan RF *chain* = 12 memiliki kinerja BER yang paling baik diikuti sistem dengan RF *chain* = 2, RF *chain* = 4, dan RF *chain* = 6 kanal URLoS.



Gambar 4.14 Grafik SE sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal URLoS

Kinerja SE sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal URLoS dapat dilihat pada Gambar 4.14 dan Tabel 4.7. Berdasarkan grafik pada Gambar dan Tabel tersebut, dapat dilihat bahwa kinerja SE pada sistem dengan jumlah RF *chain* = 6 (*hybrid beamforming)* hampir menyamai kinerja sistem dengan jumlah RF *chain* = 12 (*full-digital beamforming*). Sementara sistem dengan jumlah RF *chain* = 4 dan RF *chain* = 2 memiliki perbedaan kinerja SE yang cukup signifikan dibandingkan dengan sistem dengan RF *chain* = 12 dan RF *chain* = 6. Hal tersebut dapat dilihat untuk mencapai kinerja SE 20 bits/s/Hz, sistem dengan jumlah RF *chain* = 12 memiliki kinerja SNR 2 dB, sistem dengan jumlah RF *chain* = 6 memiliki kinerja SNR 4 dB, sistem dengan jumlah RF *chain* = 4 memiliki kinerja SNR 8 dB, dan sistem dengan jumlah RF *chain* = 2 memiliki kinerja SNR 26 dB. Pada sistem dengan kinerja SNR 20 dB, sistem dengan jumlah RF chain = 6 memiliki kinerja SE 13,12% lebih rendah dibandingkan dengan jumlah RF *chain* = 12, sistem dengan jumlah RF *chain* = 4 memiliki kinerja SE 62,58% lebih rendah, dan sistem dengan jumlah RF *chain* = 2 memiliki kinerja SE 248,84% lebih rendah. Sehingga dapat disimpulkan bahwa

kinerja SE sistem dengan metode *hybrid beamforming* dapat menyamai metode *full-digital beamforming* dengan menggunakan RF chain setengah dari yang digunakan pada sistem dengan metode *full-digital beamforming.* Sebaliknya, sistem dengan RF *chain* = 4 dan RF *chain* = 2 memiliki kinerja SE yang jauh lebih buruk dibandingkan dengan sistem yang memiliki jumlah RF *chain* = 6 pada kanal URLoS.

Tabel 4.7 SE pada sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal URLoS

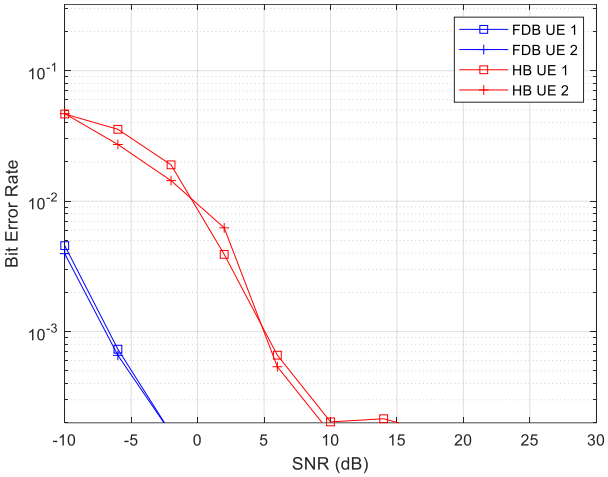
SNR (dB)	**Efisiensi Spektral (bits/s/Hz) URLoS**			
	Jumlah RF *chain*			
	12	**6**	**4**	**2**
-30	0,0520	0,0223	0,0294	0,0215
-26	0,0362	0,0853	0,1209	0,0502
-22	0,0931	0,1436	0,2022	0,1268
-18	0,2739	0,4678	0,4679	0,2823
-14	0,9132	1,0365	0,9766	0,6320
-10	2,6727	2,0214	1,9214	1,3373
-6	5,5667	5,4019	4,6204	2,5046
-2	11,6837	10,6673	8,1899	4,0646
2	20,8800	17,3473	12,5751	5,9965
6	27,2497	23,9810	17,2067	8,4376
10	35,9490	31,7806	22,1121	10,3052
14	43,4408	39,5346	27,4339	12,8015
18	51,8416	47,3976	32,8326	15,4427
22	59,8869	55,5323	38,0685	18,1162
26	68,2705	63,6129	43,3857	20,7441
30	75,9344	71,3037	48,6064	23,3361

4.5 Simulasi sistem *point-to-point* LSA-MIMO dengan metode FDB dan HB untuk 2 *user* pada kanal IRF dan URLoS

Setelah mencoba beberapa variasi konfigurasi pada sistem *point-to-point* LSA-MIMO dengan kanal IRF dan URLoS, pada subbab ini akan disimulasikan sistem sistem *point-to-point* LSA-MIMO untuk dua *user* yang berbeda dan ditransmisikan dalam waktu yang bersamaan. Perbedaan kedua user tersebut adalah sudut datang antar pamancar dengan penerima pertama dan penerima kedua. Penerima pertama berada

pada sudut 6° dari *boresight* antena pemancar, sementara penerima kedua berada pada sudut 60° dari *boresight* antena pemancar.

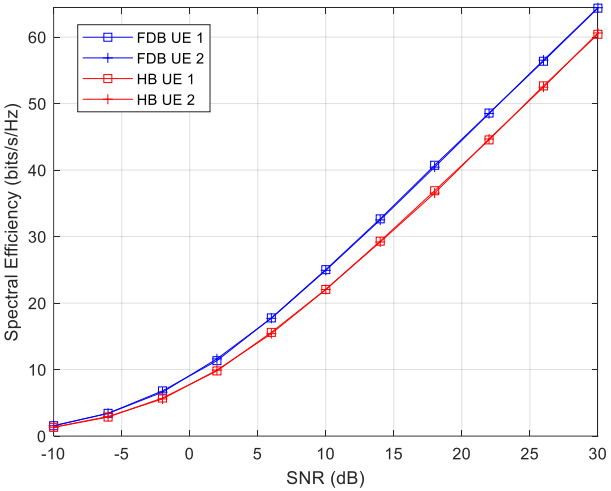
4.5.1 Hasil simulasi sistem *point-to-point* LSA-MIMO dengan metode HB dan FDB untuk 2 UE pada kanal IRF



Gambar 4.15 Grafik BER sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF

BER pada hasil simulasi sistem *point-to-point* LSA-MIMO dengan metode FDB dan HB dapat dilihat pada Gambar 4.15. Seperti yang dapat diamati pada Gambar 4.15, kedua UE tersebut memiliki kinerja BER sistem yang menyerupai satu sama lain. Pada SNR rendah, yaitu dibawah 18 dB, sistem *point-to-point* LSA-MIMO pada kanal IRF dengan metode HB memiliki kinerja BER yang lebih baik dibandingkan dengan metode FDB. Pada SNR diatas -18 dB, kinerja yang cukup signifikan terjadi pada sistem dengan metode FDB. Terlihat untuk mendapatkan BER 10^{-3}, sistem dengan metode FDB memiliki kinerja SNR -7 dB, sedangkan sistem dengan metode HB memiliki kinerja SNR 5 dB. Terdapat perbedaan 11 dB pada kedua metode untuk mencapai kinerja BER 10^{-3}. Kinerja BER sistem antara UE 1 dengan UE 2 juga tidak terlihat perbedaan yang signifikan. Pada sistem yang memiliki

kinerja SNR -10 dB, sistem berbasis FDB untuk UE 1 memiliki kinerja BER 0.004563 dan pada UE 2 memiliki kinerja BER 0.003962. Hal terserbut juga terjadi pada sistem berbasis HB. Pada SNR -10 dB, sistem untuk UE 1 memiliki BER sebesar 0.0466. Sementara sistem untuk UE 2 memiliki BER sebesar 0.0469. Maka dapat disimpulkan untuk SNR lebih dari -18 dB, sistem dengan metode FDB memiliki kinerja yang lebih baik pada sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF. Serta, tidak terdapat perbedaan yang cukup signifikan pada kinerja sistem *point-to-point* LSA-MIMO untuk 2 UE.



Gambar 4.16 Grafik SE sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF

Analisa SE pada sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF dapat dilakukan dengan melihat hasil simulasi sistem pada Gambar 4.16 dan Tabel 4.8. Dari hasil simulasi, terlihat bahwa tidak terjadi perbedaan kinerja SE untuk 2 UE tersebut. Pada SNR 10 dB, sistem *point-to-point* LSA-MIMO berbasis FDB untuk UE 1 memiliki kinerja SE sebesar 25.0266 bits/s/Hz dan sistem untuk UE 2 memiliki kinerja SE sebesar 24.9224 bits/s/Hz. Pada SNR 10 dB, sistem *point-to-point* LSA-MIMO berbasis HB untuk UE 1 memiliki kinerja SE sebesar 22.0550 bits/s/Hz, sedangkan sistem untuk UE 2 memiliki kinerja SE

sebesar 22.0397 bits/s/Hz. Namun, sedikit terdapat penurunan kinerja sistem untuk UE 2 dibandingkan dengan UE 1, jika diambil referensi pada SNR 10 dB, terdapat penurunan sebesar 0.42% pada sistem berbasis FDB dan penurunan sebesar 0.07% pada sistem berbasis HB. Untuk mendapatkan kinerja SE yang sama, semisal 30 bits/s/Hz, sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF dengan metode FDB memiliki kinerja SNR 13 dB, sementara sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF dengan metode HB memiliki kinerja SNR 14 dB. Terdapat perbedaan kinerja SNR sekitar 1 dB pada kedua metode untuk mencapai kinerja SE yang sama untuk sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF.

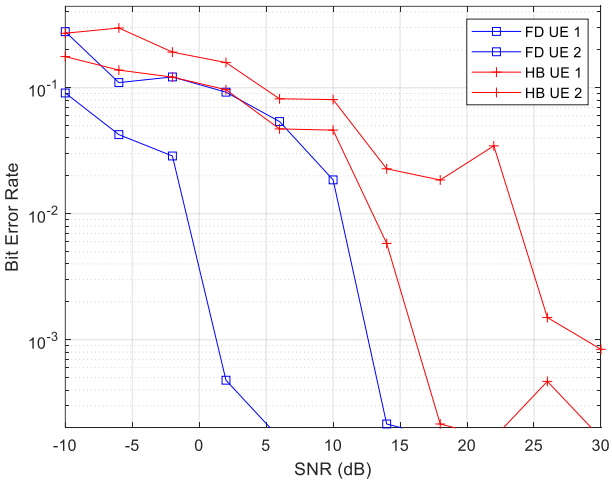
Tabel 4.8 SE pada sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF

SNR (dB)	Efisiensi Spektral (bits/s/Hz) sistem untuk 2 UE pada kanal IRF			
	Metode			
	FDB		HB	
	UE 1	UE 2	UE 1	UE 2
-10	1.5576	1.5564	1.3144	1.3343
-6	3.4244	3.4124	2.8620	2.9399
-2	6.7971	6.6055	5.7254	5.5875
2	11.3418	11.6148	9.7978	9.9187
6	17.7684	17.7131	15.5953	15.4019
10	25.0266	24.9224	22.0550	22.0397
14	32.6644	32.5066	29.3115	29.1889
18	40.7245	40.4499	36.8917	36.5412
22	48.5708	48.4974	44.5378	44.6734
26	56.3653	56.5289	52.6646	52.4688
30	64.3499	64.4170	60.4202	60.6155

4.5.2 Hasil simulasi sistem *point-to-point* LSA-MIMO dengan metode FDB dan HB untuk 2 *user* pada kanal URLoS

Pada subbab ini, analisa akan dilakukan pada hasil simulasi sistem *point-to-point* LSA-MIMO dengan membandingkan FDB dan HB untuk 2 UE pada kanal URLoS. Hasil simulasi kinerja BER dapat dilihat pada Gambar 4.17. Berdasarkan gambar 4.17, kinerja BER pada UE 1 dan UE 2 memiliki perbedaan yang sangat signifikan. Untuk mencapai kinerja

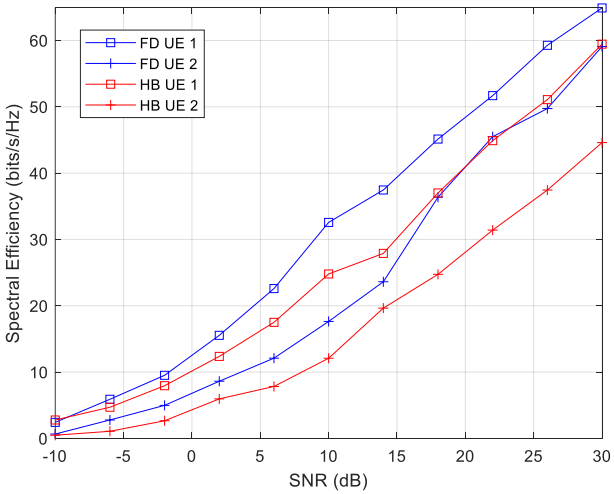
BER 10^{-3}, sistem dengan metode FDB untuk UE 1 memiliki kinerja SNR 1 dB, sistem dengan metode FDB untuk UE 2 memiliki kinerja SNR 13 dB, sistem dengan metode HB untuk UE 1 memiliki kinerja SNR 16 dB, dan sistem dengan metode HB untuk UE 2 memiliki kinerja SNR 28 dB. Pada SNR 10 dB, sistem dengan metode FDB untuk UE 1 memiliki kinerja BER 0.0002, sistem dengan metode FDB untuk UE 2 memiliki kinerja BER 0.01859, sistem dengan metode HB untuk UE 1 memiliki kinerja BER 0.04613, dan sistem dengan metode HB untuk UE 2 memiliki kinerja BER 0.08063.



Gambar 4.17 Grafik BER sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal URLoS

Analisa terhadap kinerja SE dilakukan dengan melihat hasil simulasi pada Gambar 4.18 dan Tabel 4.9. Berdasarkan hasil simulasi, terdapat perbedaan kinerja SE antara sistem *point-to-point* LSA MIMO untuk UE 1 dan UE 2. Pada SNR 10 dB, sistem dengan metode FDB untuk UE 1 memiliki kinerja SE sebesar 32.57 bits/s/Hz, sistem dengan metode FDB untuk UE 2 memiliki kinerja SE sebesar 17.64 bits/s/Hz, sistem dengan metode HB untuk UE 1 memiliki kinerja SE sebesar 24.79 bits/s/Hz, dan sistem dengan metode HB untuk UE 2 memiliki kinerja SE sebesar 12.09 bits/s/Hz. Pada sistem dengan metode FDB untuk UE 2,

terdapat penurunan kinerja sebesar 45.84% terhadap kinerja sistem untuk UE 1, sedangkan sistem dengan metode HB untuk UE 2, terdapat penurunan kinerja sebesar 51.23% dibandingkan dengan kinerja sistem untuk UE 1. Jika dibandingakan untuk UE yang sama pada metode yang berbeda, UE 1 dengan metode HB memiliki penurunan kinerja SE sebesar 23.9% dibandingkan dengan kinerja sistem untuk UE 1 dengan metode FDB. UE 2 dengan metode HB memiliki penurunan SE sebesar 31.46% jika dibandingkan dengan kinerja SE sistem untuk UE 1 dengan metode FDB.



Gambar 4.18 Grafik SE sistem point-to-point LSA MIMO untuk 2 UE pada kanal URLoS

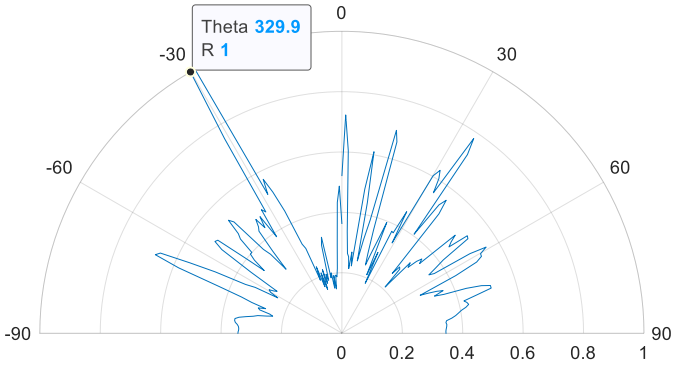
Tabel 4.9 SE pada sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal URLoS

SNR (dB)	Efisiensi Spektral (bits/s/Hz) sistem untuk 2 UE pada kanal URLoS			
	Metode			
	FDB		HB	
	UE 1	UE 2	UE 1	UE 2
-10	2.4201	0.6597	2.7783	0.4793

-6	5.9009	2.7784	4.6991	1.0733
-2	9.5157	4.9833	7.9474	2.6615
2	15.5418	8.6218	12.3686	5.9613
6	22.5900	12.0994	17.5023	7.8320
10	32.5700	17.6375	24.7944	12.0901
14	37.4581	23.6184	27.8972	19.6435
18	45.1346	36.3676	37.0090	24.7404
22	51.6843	45.4981	44.8973	31.4098
26	59.2725	49.7384	51.0809	37.4505
30	64.9060	59.0872	59.4470	44.6051

4.6 Simulasi Pola Radiasi sistem *Point-to-Point* LSA-MIMO berbasis FDB pada kanal IRF dan URLoS

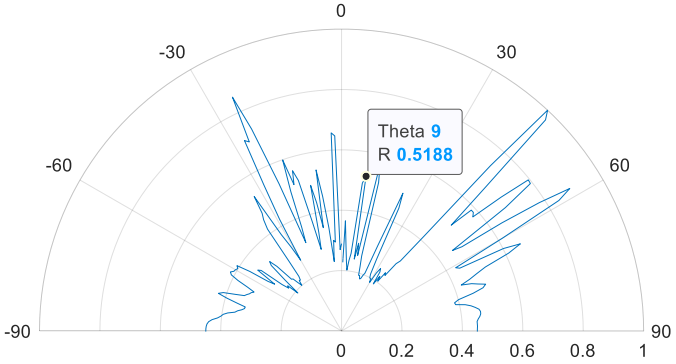
Pada subbab ini akan ditampilkan hasil simulasi pola radiasi terhadap sistem *point-to-point* MIMO berbasis FDB pada kanal IRF dan URLoS untuk melihat *beam* sinyal yang terbentuk. Antena jamak dengan konfigurasi ULA diposisikan secara horizontal sehingga *beam* sinyal yang terbentuk akan tersebar pada *azimuth* dengan sudut -90º sampai 90º derajat.



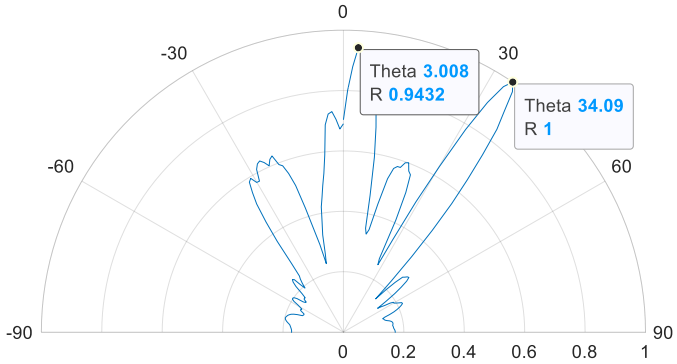
Gambar 4.19 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis FDB pada kanal IRF

Gambar 4.19 memperlihatkan pola radiasi yang dibentuk oleh BS pada sistem *point-to-point* LSA-MIMO pada kanal IRF. Pada kanal IRF, sudut pancar dari BS dipilih secara acak berdasarkan kanal yang

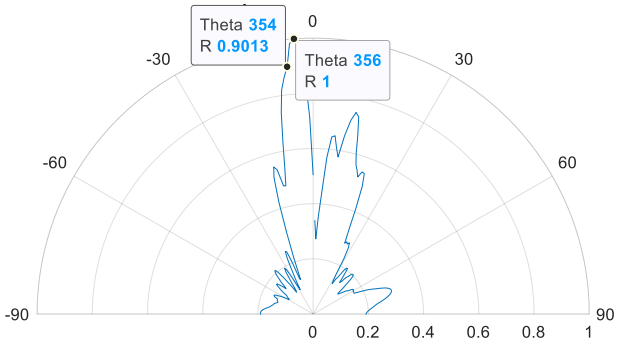
dibangkitkan. Terdapat *beam* sinyal yang mendapatkan puncak paling tinggi pada sudut sekitar -30º derajat. Sementara pada Gambar 4.20, memperlihatkan pola radiasi UE pada sistem *point-to-point* LSA-MIMO pada kanal IRF. Terlihat terdapat dua *beam* yang memiliki puncak tinggi yaitu pada sudut 3º dan 34º.



Gambar 4.21 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis FDB pada kanal URLoS



Gambar 4.20 Pola radiasi UE sistem *point-to-point* LSA-MIMO berbasis FDB pada kanal IRF



Gambar 4.22 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis FDB pada kanal URLoS

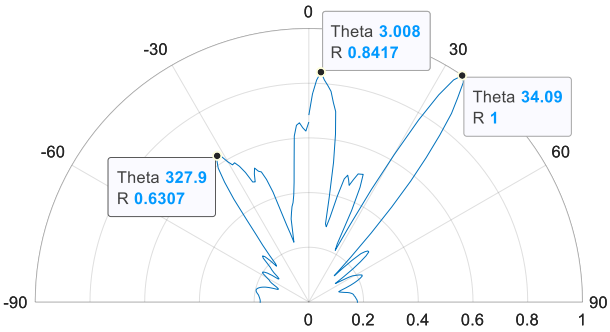
Pada kanal URLoS, sudut pancar pada BS dapat ditentukan sebesar 6º. Gambar 4.21 memperlihatkan pola radiasi BS pada sistem *point-to-point* LSA-MIMO berbasis HB. Terlihat beberapa *beam* yang memiliki puncak dengan amplitude tinggi. Terdapat peak pada sudut 9o yang merupakan puncak terdekat dengan 6º. Pola radiasi UE sistem *point-to-point* LSA-MIMO berbasis HB pada kanal URLoS dapat dilihat pada Gambar 4.22, dimana terlihat *beam* dengan puncak tertinggi terdapat pada sudut -4º namun sudut 6º juga mendapat *lobe* dari *beam* tinggi tersebut.

4.7 Simulasi Pola Radiasi sistem *Point-to-Point* MIMO berbasis HB pada kanal IRF dan URLoS

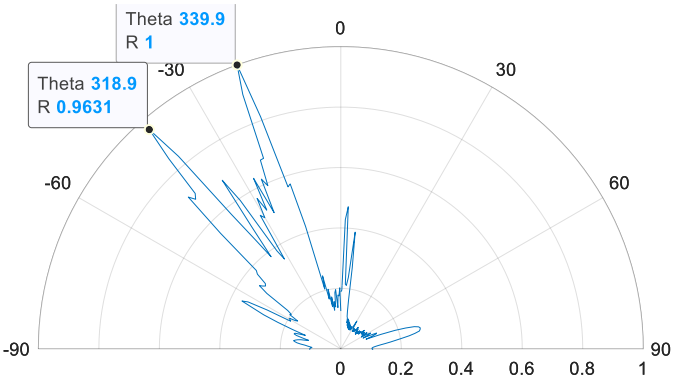
Pada subbab ini akan ditampilkan hasil simulasi pola radiasi terhadap sistem *point-to-point* MIMO berbasis HB pada kanal IRF dan URLoS untuk melihat *beam* sinyal yang terbentuk. Pada Gambar 4.23, terlihat 2 *beam* sinyal besar yang terbentuk. *Beam* dengan puncak tertinggi terdapat pada sekitar -20º dan *beam* kedua terletak pada sudut sekitar -41º.

Pola radiasi yang diterima oleh UE dapat dilihat pada Gambar 4.24. Dikarenakan pada UE juga terdapat sistem *point-to-point* MIMO berbasis HB, maka *beam* sinyal dapat diarahkan menuju BS yang memancarkan sinyal menuju UE tersebut. Dari Gambar 4.24, terlihat UE mempunyai 3 *beam* dominan dimana puncak sinyal tertinggi terdapat pada sudut 34.09º, beam tertinggi kedua pada sudut 3º, dan *beam* ketiga dengan sudut 32º. *Beam* sinyal utama pada UE kurang lebih mengarah pada BS dengan terjadi sedikit perubahan sudut akibat adanya pantulan

yang juga menyebabkan perlunya *beam* tambahan untuk menerima sinyal yang dipancarkan oleh BS.



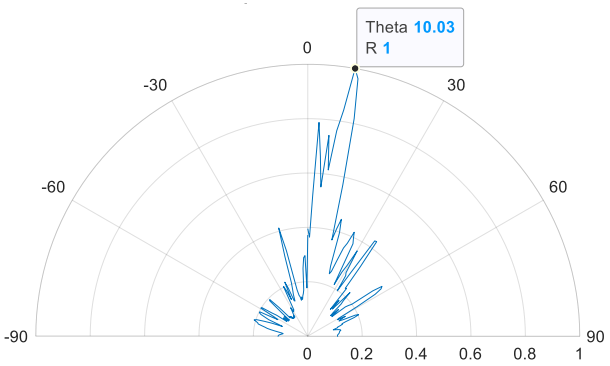
Gambar 4.23 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis HB pada kanal IRF



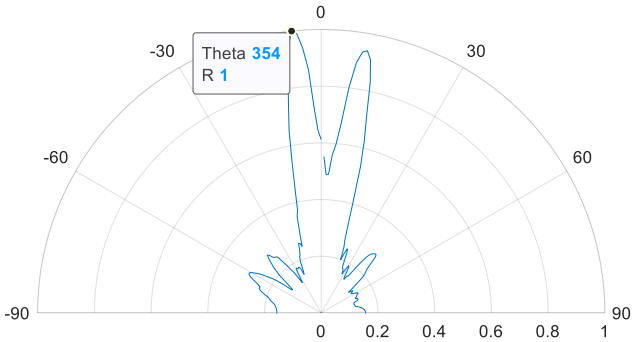
Gambar 4.24 Pola radiasi UE sistem *point-to-point* LSA-MIMO berbasis HB pada kanal IRF

Gambar 4.25 merupakan pola radiasi BS sistem *point-to-point* LSA-MIMO pada kanal URLoS. Pada kanal URLoS, sudut pancar ditentukan sebesar 6^{o}. Terdapat *beam* dengan puncak tertinggi pada sudut 10^{o} dan sudut 6^{o} juga mendapatkan *lobe* dari *beam* tersebut. Pada Gambar 4.26, memperlihatkan pola radasi UE pada sistem *point-to-point* LSA-

MIMO pada kanal URLoS. *Beam* dengan puncak terbesar terdapat pada sudut -6^{o}



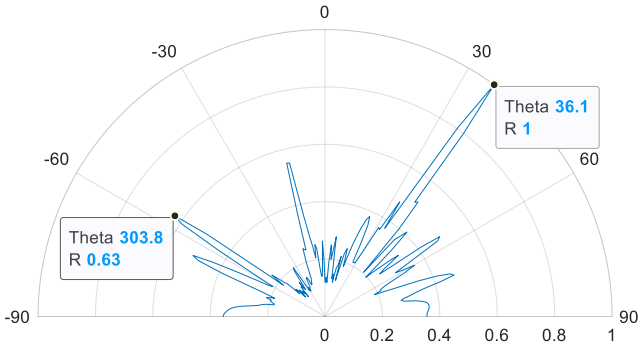
Gambar 4.25 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis HB pada kanal URLoS



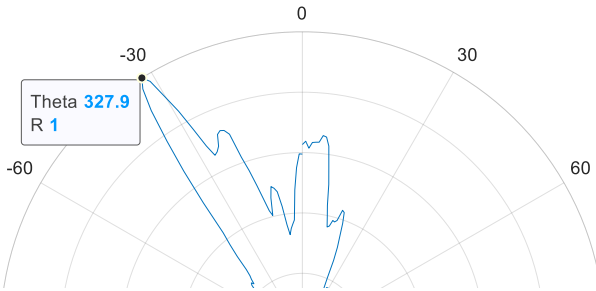
Gambar 4.26 Pola radiasi UE sistem *point-to-point* LSA-MIMO berbasis HB pada kanal URLoS

4.8 Simulasi Pola Radiasi sistem *Point-to-Point* LSA-MIMO untuk 2 UE berbasis FDB pada kanal IRF dan URLoS

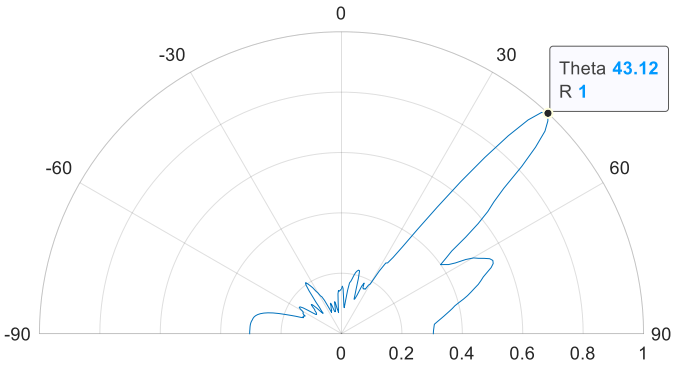
Simulasi pada subbab ini bertujuan untuk menujukkan pola radiasi yang dipancarkan oleh BS sistem *point-to-point* MIMO berbasis FDB pada kanal IRF untuk 2 UE secara bersamaan. Pada kanal IRF, sudut pancar dibangkitkan secara acak. Gambar 4.27 menunjukkan pola radiasi BS sistem *point-to-point* MIMO berbasis FDB pada kanal IRF untuk 2 UE secara bersamaan, sementara Gambar 4.28 dan Gambar 4.29 menunjukkan pola radiasi pada UE 1 dan UE 2. Pola radiasi untuk UE 1 memperlihatkan *beam* utama pada sudut 32º dan UE mempunyai *beam* utama pada sudut 43º.



Gambar 4.28 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis FDB pada kanal IRF

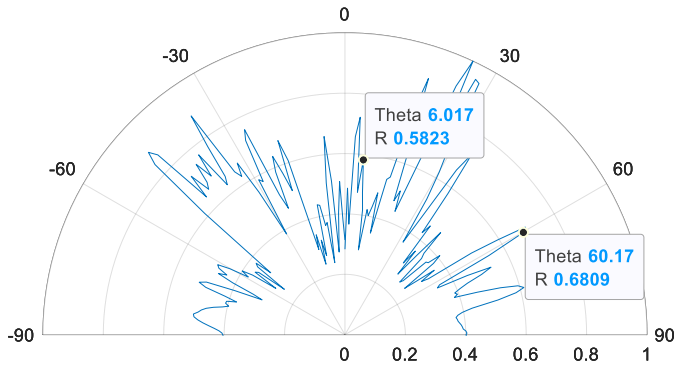


Gambar 4.27 Pola radiasi UE 1 sistem *point-to-point* LSA-MIMO berbasis FDB pada kanal IRF

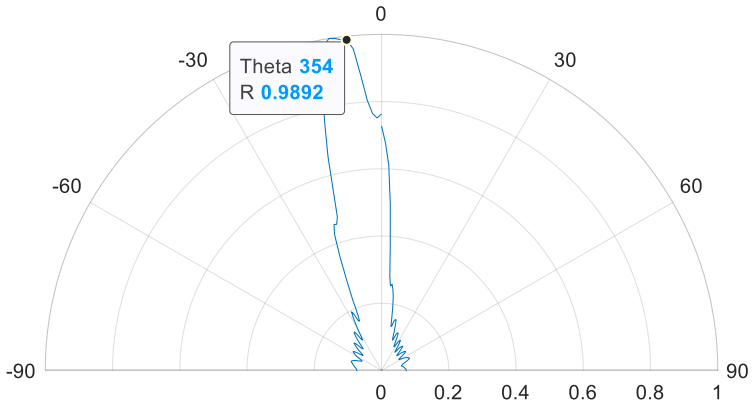


Gambar 4.30 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis FDB untuk 2 UE pada kanal IRF

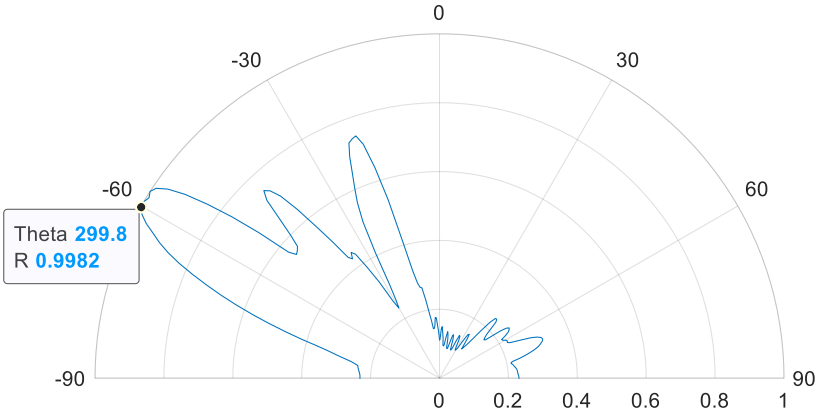
Pada Gambar 4.30, didapatkan pola radiasi BS sistem *point-to-point* MIMO berbasis FDB pada kanal URLoS untuk 2 UE secara bersamaan., terlihat beberapa *beam* yang memiliki sudut acak. Namun pada sudut yang dituju, yaitu sudut 6^o dan 60^o, terdapat *peak* pada sudut 60^o dan peak dekat dengan sudut 6^o. Gambar 4.31 dan Gambar 4.32 menunjukkan pola radiasi UE 1 dan UE 2 untuk sistem *point-to-point* MIMO berbasis FDB pada kanal IRF. Seperti yang terlihat pada pola radiasi UE, UE 1 memiliki *beam* utama pada sudut -6^o dan UE 2 memiliki *beam* utama pada sudut -60^o. Maka dapat disimpulkan bahwa kedua UE dapat memancarkan dan mendapat *beam* menuju arah yang diinginkan.



Gambar 4.29 Pola radiasi UE sistem *point-to-point* LSA-MIMO berbasis FDB pada kanal IRF



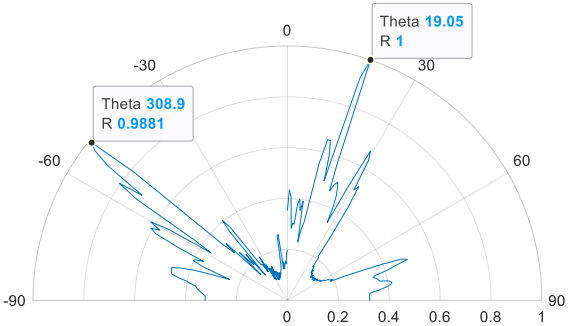
Gambar 4.32 Pola radiasi UE 1 sistem *point-to-point* LSA-MIMO berbasis FDB untuk 2 UE pada kanal IRF



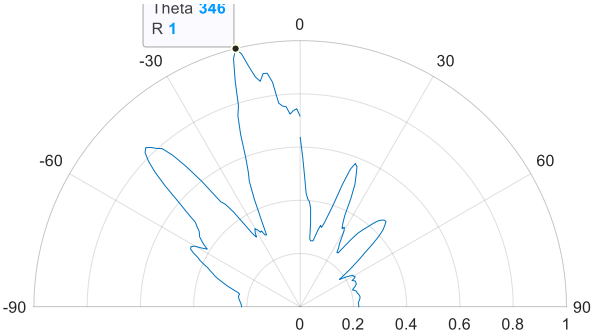
Gambar 4.31 Pola radiasi UE 2 sistem *point-to-point* LSA-MIMO berbasis FDB untuk 2 UE pada kanal IRF

4.9 Simulasi Pola Radiasi sistem *Point-to-Point* MIMO untuk 2 UE berbasis HB pada kanal IRF dan URLoS

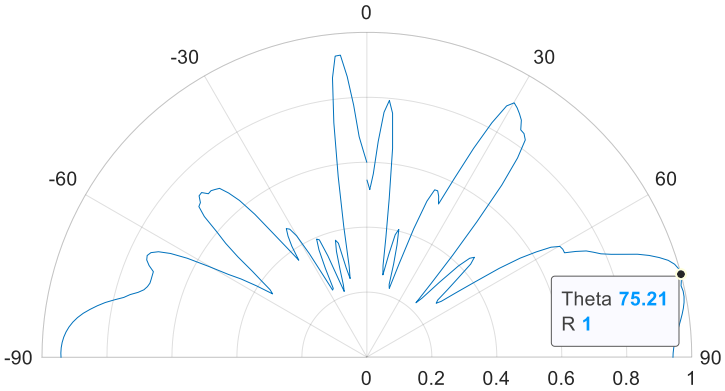
Pola radiasi BS sistem *point-to-point* MIMO berbasis FDB pada kanal IRF untuk 2 UE secara bersamaan dapat dilihat pada Gambar 4.33, dimana terlihat terdapat dua *peak* dengan puncak tertinggi pada sudut -51° dan 19°. Pada sisi penerima, pola radiasi UE dapat dilihat pada Gambar 4.34 dan Gambar 4.35. Pada pola radiasi UE 1, *peak* terbesar terlihat pada sudut 14° dan *peak* pada UE 2 terlihat pada sudut 75°. Hal ini dapat disebabkan karena pada kanal IRF, terdapat pantulan pada kanal sehingga sudut penerima berubah menyesuaikan arah pantulan sinyal tersebut.



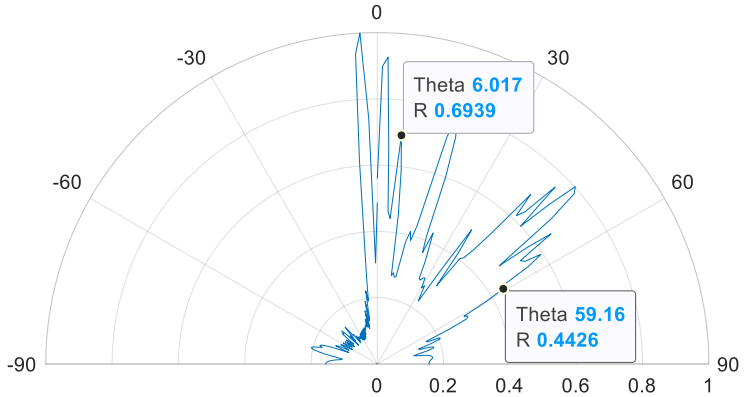
Gambar 4.34 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis HB untuk 2 UE pada kanal IRF



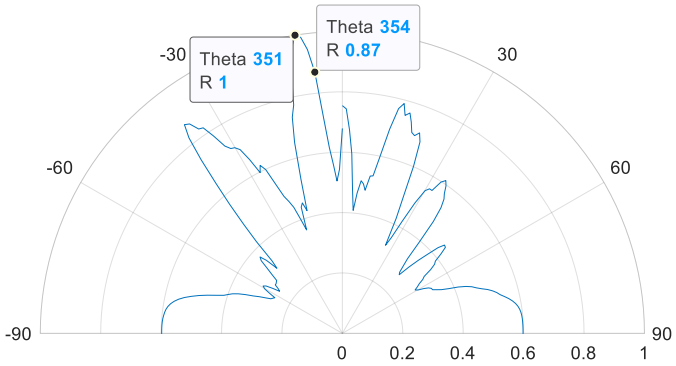
Gambar 4.33 Pola radiasi UE 1 sistem *point-to-point* LSA-MIMO berbasis HB untuk 2 UE pada kanal IRF



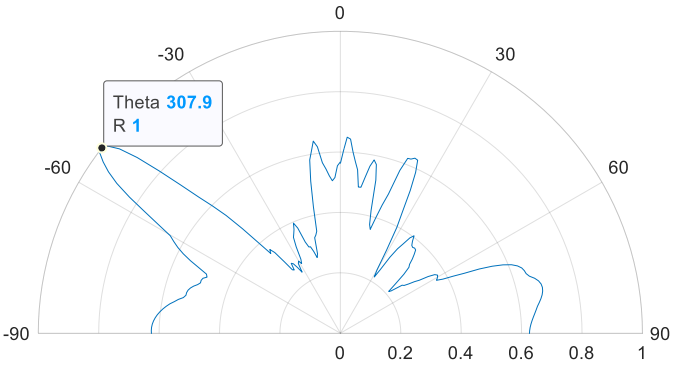
Gambar 4.35 Pola radiasi UE 2 sistem *point-to-point* LSA-MIMO berbasis HB untuk 2 UE pada kanal IRF



Gambar 4.36 Pola radiasi BS sistem *point-to-point* LSA-MIMO berbasis HB untuk 2 UE pada kanal URLoS



Gambar 4.37 Pola radiasi UE 1 sistem *point-to-point* LSA-MIMO berbasis HB untuk 2 UE pada kanal URLoS



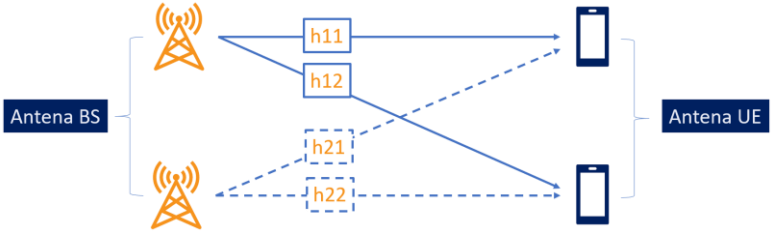
Gambar 4.38 Pola radiasi UE 2 sistem *point-to-point* LSA-MIMO berbasis HB untuk 2 UE pada kanal URLoS

Pola radiasi BS sistem *point-to-point* MIMO berbasis HB pada kanal URLoS untuk 2 UE secara bersamaan dapat dilihat pada Gambar 4.36 dan pola radiasi UE dapat dilihat pada Gambar 4.37 dan Gambar 4.38. Pada sisi pemancar, sudut pancar untuk UE 1 ditentukan sebesar 6^o dan sudut pancar untuk UE 2 ditentukan sebesar 60^o. Berdasarkan Gambar 4.40, sudut 6^o mendapatkan *beam* dengan *peak* dan pada sudut sekitar 60^o juga terdapat *beam* dengan *peak* yang cukup besar.

4.10 Diskusi

Pada perkembangan penelitian teknologi 5G, teknik *hybrid beamforming* diharapkan dapat menjadi salah satu cara untuk mencapai tujuan teknologi tersebut dikembangkan. Terutama saat penggunaan frekuensi yang semakin tinggi yang membuat sebuah keuntungan yaitu elemen antena akan semakin kecil. Pada teknologi sebelumnya, sudah dikembangkan teknik *full-digital beamforming* yang dapat mengarahkan beam menuju arah yang diinginkan secara digital. Namun, penggunaan antena dalam jumlah besar secara bersamaan dapat mengakibatkan pemborosan pada daya yang digunakan karen masing-masing elemen antena harus mempunyai RF *chain* masing-masing pada teknik *full-digital beamforming.*

Melalui Tugas Akhir ini, dibahas mengenai teknik *hybrid beamforming* yang diimplementasikan pada sistem *point-to-point* LSA-MIMO. Sebagai parameter pengukuran, digunakan dua kanal yang berbeda secara jalur transmisi, yaitu *independent Rayleigh fading* (IRF) dan *uniformly random line-of-sight* (URLoS) yang merupakan bagian dari *favorable propagation channel*. Dimana pada kanal IRF terdapat hamburan isotropis dan kanal tidak memiliki jalur *line-of-sight* antara *base station* (BS) dan *user equipment* (UE). Sementara kanal URLoS tidak memiliki hamburan serta antara BS dan UE memiliki *line-of-sight.* Berdasarkan hasil simulasi yang telah dianalisa, perbedaan kinerja antara kedua kanal tersebut terlihat pada saat dilakukan variasi terhadap jumlah antena pemancar (Tx) pada BS dan antena penerima (Rx) pada UE. Pada kanal IRF, perubahan terhadap jumlah antena Tx dan Rx tidak berpengaruh signifikan terhadap kinerja SE maupun BER sistem. Sedangkan, pada kanal URLoS, variasi jumlah antena Tx dan Rx berpengaruh terhadap kinerja SE dan BER. Semakin besar nilai antena yang digunakan pada kedua sisi Tx dan Rx, maka semakin besar pula kinerja SE dan BER yang dihasilkan.



Gambar 4.39 Ilustrasi korelasi kanal antara BS dengan UE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	0.1086	0.1736	0.0865	-0.0321	0.1007	-0.0494	-0.0079	0.0597	0.0354	0.1090	0.0522	0.1816	-0.0251	-0.0738	-0.0702
2	0.1086	1	0.0310	0.0967	-0.0287	-0.0306	-0.0287	0.0118	0.1189	-0.0948	0.0245	-0.0407	-0.1204	-0.0396	0.0633	-0.1896
3	0.1736	0.0310	1	-0.1612	0.1888	-0.0597	-0.1136	0.1459	0.0319	-0.0814	-0.0120	-0.0595	0.0202	0.0152	-0.0920	0.0810
4	0.0865	0.0967	-0.1612	1	-0.1291	0.1173	-0.0157	0.0182	0.0470	-0.0219	-0.0786	0.0425	0.0687	0.0871	-0.0796	-0.0931
5	-0.0321	-0.0287	0.1888	-0.1291	1	0.1076	-0.1524	0.0915	-0.2043	-0.1428	-0.0051	0.1148	0.0133	-0.0180	-0.0615	-0.0299
6	0.1007	-0.0306	-0.0597	0.1173	0.1076	1	-0.0129	-0.0533	-0.0113	-0.1153	-0.0769	0.0786	0.0562	-0.1324	-0.0541	0.0576
7	-0.0494	-0.0287	-0.1136	-0.0157	-0.1524	-0.0129	1	0.1298	0.0347	0.1710	0.0418	-0.0867	-0.0294	0.0138	0.1511	-0.1442
8	-0.0079	0.0118	0.1459	0.0182	0.0915	-0.0533	0.1298	1	0.0284	-0.1023	-0.0298	-0.1028	-0.0832	0.0340	-0.0591	-0.0126
9	0.0597	0.1189	0.0319	0.0470	-0.2043	-0.0113	0.0347	0.0284	1	0.0401	0.0455	0.1199	-0.0851	0.0582	-0.0457	0.0632
10	0.0354	-0.0948	-0.0814	-0.0219	-0.1428	-0.1153	0.1710	-0.1023	0.0401	1	-0.0704	-0.2467	0.0140	0.0812	0.0795	0.0747
11	0.1090	0.0245	-0.0120	-0.0786	-0.0051	-0.0769	0.0418	-0.0298	0.0455	-0.0704	1	0.0858	0.1253	-0.0854	-0.0029	-0.1372
12	0.0522	-0.0407	-0.0595	0.0425	0.1148	0.0786	-0.0867	-0.1028	0.1199	-0.2467	0.0858	1	0.0315	0.0039	-0.0611	0.0213
13	0.1816	-0.1204	0.0202	0.0687	0.0133	0.0562	-0.0294	-0.0832	-0.0851	0.0140	0.1253	0.0315	1	0.0241	-0.0968	-0.0303
14	-0.0251	-0.0396	0.0152	0.0871	-0.0180	-0.1324	0.0138	0.0340	0.0582	0.0812	-0.0854	0.0039	0.0241	1	0.0814	-0.0349
15	-0.0738	0.0633	-0.0920	-0.0796	-0.0615	-0.0541	0.1511	-0.0591	-0.0457	0.0795	-0.0029	-0.0611	-0.0968	0.0814	1	-0.1311
16	-0.0702	-0.1896	0.0810	-0.0931	-0.0299	0.0576	-0.1442	-0.0126	0.0632	0.0747	-0.1372	0.0213	-0.0303	-0.0349	-0.1311	1

Gambar 4.40 Korelasi kanal IRF

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	-0.8812	0.3777	0.3959	-0.3183	0.4826	0.2333	0.0031	-0.0519	0.0264	0.1140	-0.0309	0.2074	0.2016	-0.1630	0.1766
2	-0.8812	1	0.0022	-0.7427	0.0670	-0.5889	-0.4744	-0.1540	0.1977	0.0651	-0.0275	0.0629	-0.1626	-0.2025	0.0993	-0.1665
3	0.3777	0.0022	1	-0.5536	-0.9018	0.3439	-0.2226	-0.4704	-0.1100	0.1330	0.0081	0.0012	0.0223	0.0084	-0.1704	0.1326
4	0.3959	-0.7427	-0.5536	1	0.4822	0.4538	0.7805	0.5942	-0.3371	-0.3998	-0.2259	0.1266	-0.0171	0.0497	0.0164	0.0517
5	-0.3183	0.0670	-0.9018	0.4822	1	-0.4958	0.2285	0.7015	0.2616	-0.2785	0.0049	0.2229	0.0556	0.0079	0.0333	-0.0743
6	0.4826	-0.5889	0.3439	0.4538	-0.4958	1	0.6926	0.0068	-0.8175	-0.3546	-0.5058	0.0716	-0.3122	-0.1806	0.0685	0.0070
7	0.2333	-0.4744	-0.2226	0.7805	0.2285	0.6926	1	0.6974	-0.7415	-0.8062	-0.7006	0.4883	-0.3919	-0.3197	0.0312	-0.0624
8	0.0031	-0.1540	-0.4704	0.5942	0.7015	0.0068	0.6974	1	-0.1512	-0.8069	-0.4250	0.7311	-0.1206	-0.1924	-0.2242	0.0562
9	-0.0519	0.1977	-0.1100	-0.3371	0.2616	-0.8175	-0.7415	-0.1512	1	0.6089	0.8666	-0.2578	0.7528	0.6229	-0.3663	0.3765
10	0.0264	0.0651	0.1330	-0.3998	-0.2785	-0.3546	-0.8062	-0.8069	0.6089	1	0.8486	-0.8741	0.5604	0.6057	0.0831	0.1589
11	0.1140	-0.0275	0.0081	-0.2259	0.0049	-0.5058	-0.7006	-0.4250	0.8666	0.8486	1	-0.6190	0.8869	0.8610	-0.3058	0.4984
12	-0.0309	0.0629	0.0012	0.1266	0.2229	0.0716	0.4883	0.7311	-0.2578	-0.8741	-0.6190	1	-0.3356	-0.4915	-0.4107	0.0532
13	0.2074	-0.1626	0.0223	-0.0171	0.0556	-0.3122	-0.3919	-0.1206	0.7528	0.5604	0.8869	-0.3356	1	0.9659	-0.6345	0.8092
14	0.2016	-0.2025	0.0084	0.0497	0.0079	-0.1806	-0.3197	-0.1924	0.6229	0.6057	0.8610	-0.4915	0.9659	1	-0.5024	0.7747
15	-0.1630	0.0993	-0.1704	0.0164	0.0333	0.0685	0.0312	-0.2242	-0.3663	0.0831	-0.3058	-0.4107	-0.6345	-0.5024	1	-0.9016
16	0.1766	-0.1665	0.1326	0.0517	-0.0743	0.0070	-0.0624	0.0562	0.3765	0.1589	0.4984	0.0532	0.8092	0.7747	-0.9016	1

Gambar 4.41 Korelasi kanal URLoS

Berdasarkan definisi dari *favorable propagation channel*, perbandingan respon kanal terhadap magnitudonya akan semakin mendekati nol untuk jumlah antena menuju tidak berhingga [19]. Serta disebutkan juga bahwa kanal IRF dapat mencapai *favorable propagation channel* lebih sering dibandingkan dengan kanal URLoS [16]. Hal tersebut dapat terjadi karena pada kanal IRF, sinyal yang dipancarkan oleh masing-masing antena pemancar memiliki sifat yang independen, atau dengan kata lain kanal yang dilewat oleh antena Tx pertama tidak akan berpengaruh terhadap antena Tx kedua dan sebaliknya. Sementara pada kanal URLoS, kanal yang dilewati oleh antena Tx pertama akan perpengaruh terhadap kanal yang dilewati oleh antena Tx kedua dan juga

sebaliknya. Sehingga dimungkinkan terjadi interferensi sinyal terhadap kanal yang dilewati oleh antena Tx pada kanal URLoS. Jumlah antena yang lebih banyak juga akan membuat daya pancar semakin kuat, sehingga dapat mengurangi interferensi sinyal antar kanal yang terjadi pada kanal URLoS. Hal tersebut dibuktikan dengan matriks korelasi kanal yang terlihat pada Gambar 4.40 untuk kanal IRF dan Gambar 4.41 untuk kanal URLoS. Pada matriks korelasi, matriks diagonal merupakan autokorelasi terhadap kanal itu sendiri. Kolom dan baris pada matriks menandakan hubungan antara kanal satu dengan kanal lainnya. Pada matriks korelasi untuk kanal IRF, elemen matriks yang dekat dengan matriks diagonal memiliki nilai elemen sangat kecil jika dibandingkan dengan nilai elemen pada matriks diagonal. Sedangkan pada matriks korelasi kanal URLoS, nilai elemen dekat dengan nilai diagonal matriks mendekati nilai autokorelasi pada diagonal matriks, serta semakin jauh nomor kanal maka semakin kecil korelasi yang terjadi.

Tabel 4.10 Ringkasan hasil simulasi efisiensi spektral sistem point-to-point LSA-MIMO

Konfigurasi	Ringkasan Simulasi
Simulasi sistem *point-to-point* LSA-MIMO berbasis HB pada kanal URLoS dan IRF	Ketika kedua kanal memiliki nilai SNR 10 dB, kanal IRF mempunyai SE sebesar 22,0621 bits/s/Hz sementara kanal URLoS mempunyai SE sebesar 31,9902 bits/s/Hz pada nilai SNR yang sama.
Simulasi sistem *point-to-point* LSA-MIMO berbasis FDB dan HB pada kanal IRF	Pada SNR 10 dB, kanal IRF dengan metode FDB mempunyai SE sebesar 26,04 bits/s/Hz, sedangkan pada metode HB, SE yang didapat adalah 22,11 bits/s/Hz. Terdapat peningkatan SE sebesar 3,93 bits/s/Hz pada metode FDB jika dibandingkan dengan metode HB, atau meningkat sebesar 17%.
Simulasi sistem *point-to-point* LSA-MIMO berbasis FDB dan HB pada kanal URLoS	Saat SNR 10 dB, metode FDB mempunyai kinerja SE 36,08 bits/s/Hz dan metode HB mempunyai kinerja SE 32,04 bits/s/Hz. Oleh sebab itu, pada kanal URLoS, kinerja SE metode HB dapat menyerupai kinerja SE metode

	FDB dengan selisih 4,04 bits/s/Hz. Terjadi peningkatan sebesar 12,61% untuk sistem FDB.
Simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal IRF	Saat SNR memiliki nilai 10 dB, untuk Tx=64 memiliki kinerja SE sebesar 22,6 bits/s/Hz, untuk Tx=32 memiliki kinerja SE sebesar 21,43 bits/s/Hz, untuk Tx=16 memiliki kinerja SE sebesar 20,61 bits/s/Hz.
Simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal IRF	Pada kinerja SNR 40 dB, sistem dengan jumlah Rx=16 memiliki kinerja SE 47,83 bits/s/Hz, sistem dengan jumlah Rx=12 memiliki kinerja SE 47,55 bits/s/Hz, dan sistem dengan jumlah Rx=8 memiliki kinerja SE 46,35 bits/s/Hz.
Simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah Tx pada kanal URLoS	Pada kinerja SNR 10 dB, sistem dengan jumlah antena Tx=64 memiliki kinerja SE 31,41 bits/s/Hz, sistem dengan jumlah antena Tx=32 memiliki kinerja SE 20,67 bits/s/Hz, dan sistem dengan jumlah antena Tx=16 memiliki kinerja SE 13,72 bits/s/Hz.
Simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah Rx pada kanal URLoS	Pada SNR 10 dB, sistem dengan jumlah Rx=16 memiliki kinerja SE 31,64 bits/s/Hz, sistem dengan jumlah Rx=12 memiliki kinerja SE 28,89 bits/s/Hz, dan sistem dengan jumlah antena Rx=8 memiliki kinerja SE 24,02 bits/s/Hz.
Simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal IRF	Pada SNR 10 dB, sistem dengan metode HB (RF *chain* = 6) memiliki kinerja SE 17,85% lebih rendah dibandingkan dengan metode HB, sistem dengan RF *chain* = 4 memiliki kinerja SE 58,78% lebih rendah, dan sistem dengan RF

	chain = 2 memiliki kinerja SE 175,6% lebih rendah.
Simulasi sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal URLoS	Pada kinerja 10 dB, sistem dengan jumlah RF *chain* = 6 memiliki kinerja SE 13,12% lebih rendah dibandingkan dengan metode FD, sistem dengan jumlah RF *chain* = 4 memiliki kinerja SE 62,58% lebih rendah, dan sistem dengan jumlah RF *chain* = 2 memiliki kinerja SE 248,84% lebih rendah pada kanal URLoS.
Simulasi sistem *point-to-point* LSA-MIMO dengan metode FDB untuk 2 UE pada kanal IRF	Pada SNR 10 dB, sistem *point-to-point* LSA-MIMO berbasis FDB untuk UE 1 memiliki kinerja SE sebesar 25.0266 bits/s/Hz dan sistem untuk UE 2 memiliki kinerja SE sebesar 24.9224 bits/s/Hz.
Simulasi sistem *point-to-point* LSA-MIMO dengan metode HB untuk 2 UE pada kanal IRF	Pada SNR 10 dB, sistem *point-to-point* LSA-MIMO berbasis HB untuk UE 1 memiliki kinerja SE sebesar 22.0550 bits/s/Hz, sedangkan sistem untuk UE 2 memiliki kinerja SE sebesar 22.0397 bits/s/Hz.
Simulasi sistem *point-to-point* LSA-MIMO dengan metode FDB untuk 2 *user* pada kanal URLoS	Pada SNR 10 dB, sistem dengan metode FDB untuk UE 1 memiliki kinerja SE sebesar 32.57 bits/s/Hz, sistem dengan metode FDB untuk UE 2 memiliki kinerja SE sebesar 17.64 bits/s/Hz.
Simulasi sistem *point-to-point* LSA-MIMO dengan metode HB untuk 2 *user* pada kanal URLoS	Pada SNR 10 dB, sistem dengan metode HB untuk UE 1 memiliki kinerja SE sebesar 24.79 bits/s/Hz, dan sistem dengan metode HB untuk UE 2 memiliki kinerja SE sebesar 12.09 bits/s/Hz.

[Halaman Ini Sengaja Dikosongkan]

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisa dan pembahasan terhadap pengambilan data dan simulasi terhadap kinerja BER dan SE sistem *point-to-point* LSA-MIMO pada kanal IRF dan URLoS pada bab 4, maka kesimpulan yang didapat adalah sebagai berikut:

1. Sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming* (HB) memiliki kinerja *spectral efficiency* (SE) lebih baik pada kanal URLoS dibandingkan dengan sistem pada kanal IRF.
2. Kinerja SE sistem *point-to-point* LSA-MIMO berbasis HB dapat menyerupai kinerja sistem dengan metode *full-digital beamforming* (FDB) pada kanal IRF dan URLoS dengan menggunakan jumlah RF *chain* setengah dari yang digunakan pada metode FDB. Sistem dengan metode HB lebih direkomendasikan untuk sistem *massive* MIMO karena membutuhkan jumlah RF *chain* 50% lebih sedikit dibandingan sistem dengan metode FDB dengan kinerja SE yang menyerupai.
3. Variasi jumlah antena pemancar (Tx) tidak berpengaruh signifikan terhadap kinerja SE pada sistem *point-to-point* LSA-MIMO pada kanal IRF, namun berpengaruh signifikan pada kanal URLoS.
4. Variasi jumlah antena penerima (Rx) tidak berpengaruh signifikan pada sistem *point-to-point* LSA-MIMO pada kanal IRF, namun berpengaruh signifikan pada kanal URLoS.
5. Pada perbandingan variasi jumlah RF *chain* sistem *point-to-point* LSA-MIMO pada kanal IRF dan URLoS, sistem dengan jumlah RF *chain* kurang dari setengah jumlah RF *chain* yang digunakan metode FDB, memiliki kinerja yang jauh lebih rendah dibandingkan sengan sistem yang menggunakan RF *chain* setengah jumlah RF *chain* yang digunakan metode FDB.
6. Sistem *point-to-point* LSA-MIMO dapat melayani 2 user yang berbeda tempat pada kanal IRF baik pada kinerja SE maupun BER. Pada kanal IRF, kinerja SE dan BER sistem untuk UE 1 dan UE 2 mengalami perubahan signifikan. Namun pada kanal URLOS, kinerja SE pada UE 2 mengalami penurunan kinerja yang signifikan dibandingkan dengan kinerja SE pada UE 1.
7. Sistem *point-to-point* LSA-MIMO berbasis FDB dan HB dapat

membentuk pola radiasi dengan teknik *beamforming* menuju atau mendekati arah yang diinginkan pada sisi BS dan UE pada kanal IRF dan URLoS.

5.2 Saran

Setelah melakukan pemodelan, analisa, dan penarikan kesimpulan, penulis memberikan saran untuk pengembangan penelitian pada topik *point-to-point* LSA-MIMO sebagai berikut:

1. Pemeriksaan kinerja BER dapat dilakukan dengan memperbanyak jumlah sampel data yang dikirimkan melalui pemancar untuk mendapatkan hasil grafik BER yang lebih halus.
2. Perlu adanya penelitian lebih lanjut mengenai jenis kanal lainnya pada sistem *point-to-point* LSA-MIMO berbasis *hybrid beamforming.*
3. Perlu adanya penelitian lebih lanjut mengenai korelasi antara jumlah antena pemancar dan antena penerima terhadap kinerja SE dan BER.

[Halaman Ini Sengaja Dikosongkan]

Daftar Pustaka

[1] W. Roh, J.-Y. Seol, B. Lee, J. Lee, Y. Kim, J. Cho and K. Cheun, "Millimeter-wave Beamforming as an enabling technology for 5G cellular communications: theoretical feasibility and prototype results," *IEEE Communications Magazine,* vol. 52, no. 2, pp. 106-113, 2014.

[2] E. Ali, M. Ismail, R. Nordin and N. F. Abdulah, "Beamforming techniques for massive MIMO systems in 5G: overview, classification, and trends for future research.," *Frontiers of Information Technology & Electronic Engineering,* vol. 18, no. 6, pp. 753-772, 2017.

[3] International Telecommunication Union, "IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond," *Recommendation ITU-R M.2083-0,* vol. M Series, 2015.

[4] A. A. Zaidi, R. Baldemair, H. Tullberg, H. Bjorkegren, L. Sundstrom, J. Medbo, C. Kilinc and I. D. Silva, "Waveform and Numerology to Support 5G Services and Requirements," *IEEE Communications Magazine,* vol. 54, no. 11, pp. 90-98, 2016.

[5] T. Wang, G. Li, J. Ding, Q. Miao, J. Li and Wang Ying, "5G Spectrum: Is China Ready?," *IEEE Communications Magazine,* vol. 53, no. 7, pp. 58-65, 2015.

[6] S. Onoe, "1.3 Evolution of 5G mobile Technology Toward 2020 and Beyond," *International Solid-State Circuits Conference (ISSCC),* vol. 1, pp. 23-28, 2016.

[7] J. Litva and T. K.-Y. Lo, Digital Beamforming in Wireless Communications, Norwood: Artech House, 1996.

[8] B. D. V. Veen and K. M. Buckley, "Beamforming: A Versatile Approach to Spatial Filtering," *IEEE ASSP Magazine,* vol. 5, no. 2, pp. 4-24, 1988.

[9] M. S. Islam, T. Jessy, S. M. Hassan, K. Mondal and T. Rahman, "Suitable beamforming technique for 5G wireless communications," *International Conference on Computing, Communication and Automation (ICCCA),* 2016.

[10] A. Mezghani and R. W. Heath, Jr., "MIMO Beampattern and

Waveform Design with Low Resolution DACs," Deparment of ECE, The University of Texas, Austin, 2018.

[11] M. K. Adityo and I. Krisnadi, "Tinjauan Frekuensi 5G di Indonesia," in *Universitas Telkom Bandung*, Bandung, 2018.

[12] S. Darzi, T. S. Kiong, M. T. Islma, H. R. Soleymanpour and S. Kibria, "A memory-based gravitational search algorithm for enhancing minimum variance distortionless response beamforming," *Applied Soft Computing,* vol. 47, pp. 103-118, 2016.

[13] V. Venkateswaran and A.-J. v. d. Veen, "Analog Beamforming in MIMO Communications With Phase Shift Networks and Online Channel Estimation," *IEEE Transactions on Signal Processing,* vol. 58, no. 8, 2010.

[14] C. B. Albert and H. Chen, "Performance Comparison in Digital Beamforming Using LMS, RLS and D3LS Algorithms," *International Journal of Computer Science and Mobile Computing,* vol. 5, no. 5, pp. 221-230, 2016.

[15] Z. Cao, Q. Ma, A. B. Smolders, Y. Jiao, M. J. Wale, C. W. Oh, H. Wu and A. M. J. Koonen, "Advanced Integration Techniques on Broadband Millimeter-Wave Beam Steering for 5G Wireless Networks and Beyond," *IEEE Journal of Quantum Electronics,* vol. 52, no. 1, pp. 1-20, 2016.

[16] T. L. Marzetta, E. G. Larsson, H. Yang and H. Q. Ngo, Fundamentals of Massive MIMO, Cambridge: Cambridge University Press, 2016.

[17] A. A. Nasir, H. Mehrpouyan, R. Schober and Y. Hua, "Phase Noise in MIMO Systems: Bayesian Cramer-Rao Bounds and Soft-Input Estimation," *IEEE Transaction on Signal Processing,* vol. 61, no. 10, pp. 2675-2692, 2013.

[18] F. Sohrabi and W. Yu, "Hybrid Digital and Analog Beamforming Design for Large-Scale Antenna Arrays," *IEEE Journal of Selected Topics in Signal Processing,* vol. 10, no. 3, pp. 501-513, 2016.

[19] H. J. D. L. Santos, C. Strum and J. Pontes, Radio Systems Engineering: A Tutorial Approach, Springer, 2015.

LAMPIRAN A

Hasil Simulasi *Bit Error Rate*

Tabel 1, BER pada sistem *point-to-point* LSA-MIMO berbasis HB dengan kanal IRF dan URLoS

SNR (dB)	*Bit Error Rate*			
	Kanal			
	IRF		URLoS	
	FDB	HB	FDB	HB
-10	0,001533	0,04895	0,02833	0,1567
-6	0,0007733	0,03683	0,01699	0,1821
-2	0,0001667	0,02677	0,009363	0,1087
2	0,0001667	0,01572	0,0001667	0,06639
6	0,0001667	0,009672	0,0001667	0,05121
10	0,0001667	0,0009867	0,0001667	0,01507
14	0,0001667	0,0001667	0,0001667	0,002987
18	0,0001667	0,0001667	0,0001667	0,0001667
22	0,0001667	0,0001667	0,0001667	0,0001667
26	0,0001667	0,0001667	0,0001667	0,0001667
30	0,0001667	0,0001667	0,0001667	0,0001667

Tabel 2, BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena pemancar (Tx) pada kanal IRF

SNR (dB)	*Bit Error Rate* IRF		
	Jumlah antena pemancar (Tx)		
	64	32	16
-10	0,0545	0,0542	0,0574
-6	0,0365	0,0429	0,0394
-2	0,0233	0,0268	0,0295
2	0,0072	0,0169	0,0196
6	0,0008	0,0091	0,0100
10	0,0002	0,0015	0,0020
14	0,0002	0,0002	0,0002
18	0,0002	0,0002	0,0002

22	0,0002	0,0002	0,0002
26	0,0002	0,0002	0,0002
30	0,0002	0,0002	0,0002

Tabel 3, BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena penerima (Rx) pada kanal IRF

SNR (dB)	***Bit Error Rate* IRF**		
	Jumlah antena penerima (Rx)		
	16	**12**	**8**
-10	0,0550	0,0486	0,0467
-6	0,0382	0,0398	0,0359
-2	0,0266	0,0247	0,0258
2	0,0141	0,0175	0,0132
6	0,0053	0,0073	0,0070
10	0,0017	0,0028	0,0038
14	0,0002	0,0002	0,0002
18	0,0002	0,0002	0,0002
22	0,0002	0,0002	0,0002
26	0,0002	0,0002	0,0002
30	0,0002	0,0002	0,0002

Tabel 4, BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena pemancar (Tx) pada kanal URLoS

SNR (dB)	***Bit Error Rate* URLoS**		
	Jumlah antena pemancar (Tx)		
	64	**32**	**16**
-10	0,1553	0,2352	0,2967
-6	0,1815	0,2155	0,2633
-2	0,1134	0,1750	0,2430
2	0,0627	0,1525	0,2297
6	0,0452	0,1405	0,2346
10	0,0056	0,0954	0,1954
14	0,0008	0,0873	0,1784
18	0,0002	0,0759	0,1620
22	0,0002	0,0749	0,1589

26	0,0002	0,0620	0,1510
30	0,0002	0,0358	0,1153

Tabel 5, BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah antena penerima (Rx) pada kanal URLoS

SNR (dB)	***Bit Error Rate* URLoS**		
	Jumlah antena penerima (Rx)		
	16	**12**	**8**
-10	0,1702	0,1465	0,2302
-6	0,1895	0,1595	0,2072
-2	0,1139	0,1549	0,1744
2	0,0651	0,1037	0,1445
6	0,0490	0,0649	0,1327
10	0,0090	0,0667	0,1156
14	0,0004	0,0441	0,0831
18	0,0002	0,0084	0,0723
22	0,0002	0,0032	0,0698
26	0,0002	0,0002	0,0647
30	0,0002	0,0002	0,0187

Tabel 6, BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal IRF

SNR (dB)	***Bit Error Rate* IRF**			
	Jumlah RF *chain*			
	12	**6**	**4**	**2**
-14	0,0377	0,0693	0,0655	0,0427
-10	0,0003	0,0507	0,0514	0,0227
-6	0,0003	0,0388	0,0331	0,0026
-2	0,0002	0,0265	0,0164	0,0002
2	0,0002	0,0132	0,0052	0,0002
6	0,0002	0,0065	0,0033	0,0002
10	0,0002	0,0011	0,0002	0,0002
14	0,0002	0,0002	0,0002	0,0002
18	0,0002	0,0002	0,0002	0,0002
22	0,0002	0,0002	0,0002	0,0002
26	0,0002	0,0002	0,0002	0,0002

30	0,0002	0,0002	0,0002	0,0002

Tabel 7, BER sistem *point-to-point* LSA-MIMO dengan variasi jumlah RF *chain* pada kanal URLoS

SNR (dB)	***Bit Error Rate* IRF**			
	Jumlah RF *chain*			
	12	**6**	**4**	**2**
-10	0,0422	0,1616	0,1747	0,1375
-6	0,0124	0,1857	0,1511	0,0825
-2	0,0135	0,1073	0,0682	0,0251
2	0,0004	0,0660	0,0419	0,0053
6	0,0002	0,0496	0,0154	0,0002
10	0,0002	0,0161	0,0020	0,0002
14	0,0002	0,0062	0,0007	0,0002
18	0,0002	0,0006	0,0002	0,0002
22	0,0002	0,0003	0,0002	0,0002
26	0,0002	0,0002	0,0002	0,0002
30	0,0002	0,0002	0,0002	0,0002

Tabel 8, BER pada sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal IRF

SNR (dB)	***Bit Error Rate* sistem untuk 2 UE pada kanal IRF**			
	Metode			
	FDB		**HB**	
	UE 1	**UE 2**	**UE 1**	**UE 2**
-10	0.0046	0.0040	0.0466	0.0469
-6	0.0007	0.0007	0.0356	0.0272
-2	0.0002	0.0002	0.0190	0.0144
2	0.0002	0.0002	0.0039	0.0062
6	0.0002	0.0002	0.0007	0.0005
10	0.0002	0.0002	0.0002	0.0002
14	0.0002	0.0002	0.0002	0.0002
18	0.0002	0.0002	0.0002	0.0002
22	0.0002	0.0002	0.0002	0.0002
26	0.0002	0.0002	0.0002	0.0002
30	0.0002	0.0002	0.0002	0.0002

Tabel 9, BER pada sistem *point-to-point* LSA-MIMO untuk 2 UE pada kanal URLoS

SNR (dB)	***Bit Error Rate* sistem untuk 2 UE pada kanal URLoS**			
	Metode			
	FDB		**HB**	
	UE 1	**UE 2**	**UE 1**	**UE 2**
-10	0.0910	0.2787	0.1768	0.2710
-6	0.0425	0.1100	0.1382	0.2981
-2	0.0288	0.1219	0.1216	0.1920
2	0.0005	0.0923	0.0966	0.1586
6	0.0002	0.0540	0.0472	0.0818
10	0.0002	0.0186	0.0461	0.0806
14	0.0002	0.0002	0.0058	0.0228
18	0.0002	0.0002	0.0002	0.0185
22	0.0002	0.0002	0.0002	0.0346
26	0.0002	0.0002	0.0005	0.0015
30	0.0002	0.0002	0.0002	0.0008

LAMPIRAN B
LISTING PROGRAM

```
close all;
clear all;
clc;

%parameter tetap sistem
f = 28;          % frekuensi carrier dalam GHz
h_BS = 25;       % tinggi BS dalam meter
h_us = 1.65;     % tinggi UE dalam meter
d_mk = 200;      % jarak BS dan user dalam meter
m = 1:1:64;      % jumlah antena pada terminal
num_m = length(m); % jumlah antena ada terminal
u = 1:1:16;        % jumlah antena pada user
num_u = length(u); % jumlah antena pada user
i = sqrt(-1);

% Pamameter sistem Full-Digital Beamforming
d_FDB = 6;            % jumlah data stream
K_FDB = 1;            % jumlah dari user
Ns_FDB = K_FDB*d_FDB;
NRFt_FDB = Ns_FDB*2;  % jumlah RF chain transmiter
NRFr_FDB = Ns_FDB*2;  % jumlah RF chain receiver
NRF_FDB = Ns_FDB*2;

%         Pamameter         sistem         Hybrid
Beamforming                                      %
Power budget dari transmitter
d_HB = 6;          % data stream user
K_HB = 1;          % jumlah dari user
Ns_HB = K_HB*d_HB; % jumlah data stream
NRFt_HB = Ns_HB;   % RF chain pada transmiter
NRFr_HB = Ns_HB;   % RF chain pada receiver
NRF_HB = Ns_HB;

% Parameter TX yang diubah (jumlah antena BS = 64)
m_TX64 = 1:1:64;   % jumlah antena pada terminal
num_m_TX64 = length(m_TX64); % jumlah antena pada
```

```
terminal
u_TX64 = 1:1:16; % jumlah antena pada user
num_u_TX64 = length(u_TX64); % jumlah antena pada
user

% Parameter TX yang diubah (jumlah antena BS = 32)
m_TX32=1:1:32;  % jumlah antena pada terminal
num_m_TX32=length(m_TX32); % jumlah antena pada
terminal
u_TX32=1:1:16; % jumlah antena pada user
num_u_TX32=length(u_TX32);       % jumlah antena
pada user

% Parameter TX yang diubah (jumlah antena BS = 16)
m_TX16=1:1:16;  % jumlah antena pada terminal
num_m_TX16=length(m_TX16);%  jumlah  antena  ada
terminal
u_TX16 = 1:1:16; % jumlah antena pada user
num_u_TX16 = length(u_TX16);      % jumlah antena
pada user

% Parameter RX yang diubah (jumlah antena user =
16)
m_RX64=1:1:64;  % jumlah antena pada terminal
num_m_RX64=length(m_RX64); % jumlah antena ada
terminal
u_RX64=1:1:16;   % jumlah antena pada user
num_u_RX64=length(u_RX64);       % jumlah antena
pada user

% Parameter RX yang diubah (jumlah antena user =
12)
m_RX32=1:1:64;  % jumlah antena pada terminal
num_m_RX32=length(m_RX32);   % jumlah antena pada
terminal
u_RX32=1:1:12;  % jumlah antena pada user
num_u_RX32    = length(u_RX32);  % jumlah antena
pada user

% Parameter RX yang diubah(jumlah antena user=8)
```

```
m_RX16= 1:1:64; % jumlah antena pada terminal
num_m_RX16=length(m_RX16); % jumlah antena pada
terminal
u_RX16=1:1:8;   % jumlah antena pada user
num_u_RX16=length(u_RX16);       % jumlah antena
pada user

%         Pamameter         sistem         Hybrid
Beamforming                                        %
Power budget dari transmitter
d_HB  =  6;    % jumlah dari data stream yang
dibutuhkan masing masing user
K_HB = 1;  % jumlah dari user
Ns_HB = K_HB*d_HB;
NRFt_HB = Ns_HB;  % RF chain pada transmiter
NRFr_HB = Ns_HB;  % RF chain pada receiver
NRF_HB = Ns_HB;

%     Pamameter     sistem     dengan     4     RF
Chain                                % Power budget
dari transmitter
d_RF4  =  4;    % jumlah dari data stream yang
dibutuhkan masing masing user
K_RF4 = 1;   % jumlah dari user
Ns_RF4 = K_RF4*d_RF4;
NRFt_RF4 = 4; % RF chain pada transmiter
NRFr_RF4 = 4; % RF chain pada receiver
NRF_RF4 = 4;

% Pamameter sistem dengan 2 RF Chain
d_RF2  =  2;    % jumlah dari data stream yang
dibutuhkan masing masing user
K_RF2 = 1;  % jumlah dari user
Ns_RF2 = K_RF2*d_RF2;
NRFt_RF2 = 2;   %RF chain pada transmiter
NRFr_RF2 = 2;   %RF chain pada receiver
NRF_RF2 = 2;

% Pamameter sistem Full-Digital Beamforming untuk
user 1
```

```
d_FDB_US1 = 6; % jumlah dari data stream yang
dibutuhkan masing masing user
K_FDB_US1 = 1;        % jumlah dari user
Ns_FDB_US1 = K_FDB_US1*d_FDB_US1;
NRFt_FDB_US1 = Ns_FDB_US1*2;    % RF chain pada
transmiter
NRFr_FDB_US1 = Ns_FDB_US1*2;  %RF chain pada
receiver
NRF_FDB_US1 = Ns_FDB_US1*2;

% Pamameter sistem Full-Digital Beamforming untuk
user 2
d_FDB_US2 = 6; % jumlah dari data stream yang
dibutuhkan masing masing user
K_FDB_US2 = 1; % jumlah dari user
Ns_FDB_US2 = K_FDB_US2*d_FDB_US2;
NRFt_FDB_US2 = Ns_FDB_US2*2; % NRFt = jumlah dari
RF chain pada transmiter
NRFr_FDB_US2 = Ns_FDB_US2*2; % NRFr = number dari
RF chain pada receiver
NRF_FDB_US2 = Ns_FDB_US2*2;

% Pamameter sistem Hybrid Beamforming untuk user
1
d_HB_US1 = 6; % jumlah dari data stream yang
dibutuhkan masing masing user
K_HB_US1 = 1; % jumlah dari user
Ns_HB_US1 = K_HB_US1*d_HB_US1;
NRFt_HB_US1 = Ns_HB_US1;  % NRFt = jumlah dari RF
chain pada transmiter
NRFr_HB_US1 = Ns_HB_US1;  % NRFr = number dari RF
chain pada receiver
NRF_HB_US1 = Ns_HB_US1;

% Pamameter sistem Hybrid Beamforming untuk user
2
d_HB_US2 = 6;   % jumlah dari data stream yang
dibutuhkan masing masing user
K_HB_US2 = 1;  % jumlah dari user
Ns_HB_US2 = K_HB_US2*d_HB_US2;
```

```
NRFt_HB_US2 = Ns_HB_US2;                                    %
NRFt = jumlah dari RF chain pada transmiter
NRFr_HB_US2 = Ns_HB_US2;                                    %
NRFr = number dari RF chain pada receiver
NRF_HB_US2 = Ns_HB_US2;
total_Rx_US2 = 32;

% SNR range
SNR_dB = -10:4:30;      % SNR range in dB
SNR_Lin = 10.^(SNR_dB/10);  % SNR range in linear
power
N_sistem = 1;  % jumlah itersi kanal

SE_IRF_FDB = zeros(length(SNR_dB),N_sistem);
SE_IRF_HB = zeros(length(SNR_dB),N_sistem);
SE_URLOS_FDB = zeros(length(SNR_dB),N_sistem);
SE_URLOS_HB = zeros(length(SNR_dB),N_sistem);

SE_IRF_TX64 = zeros(length(SNR_dB),N_sistem);
SE_IRF_TX32 = zeros(length(SNR_dB),N_sistem);
SE_IRF_TX16 = zeros(length(SNR_dB),N_sistem);
SE_urlos_TX64 = zeros(length(SNR_dB),N_sistem);
SE_urlos_TX32 = zeros(length(SNR_dB),N_sistem);
SE_urlos_TX16 = zeros(length(SNR_dB),N_sistem);

SE_IRF_RX64 = zeros(length(SNR_dB),N_sistem);
SE_IRF_RX32 = zeros(length(SNR_dB),N_sistem);
SE_IRF_RX16 = zeros(length(SNR_dB),N_sistem);
SE_urlos_RX64 = zeros(length(SNR_dB),N_sistem);
SE_urlos_RX32 = zeros(length(SNR_dB),N_sistem);
SE_urlos_RX16 = zeros(length(SNR_dB),N_sistem);

SE_IRF_RF4 = zeros(length(SNR_dB),N_sistem);
SE_IRF_RF2 = zeros(length(SNR_dB),N_sistem);
SE_URLOS_RF4 = zeros(length(SNR_dB),N_sistem);
SE_URLOS_RF2 = zeros(length(SNR_dB),N_sistem);
%Ns_HB = 1;
% for SNR loop
for SNRLoop = 1 : length(SNR_dB)
```

```
    SNR = SNR_Lin(SNRLoop); % random channel
realization

% Algoritma 1 : Desain VRF untuk point-to-point
MIMO

for n_sistem = 1 : N_sistem
    P_sistem = 1;
    variance_sistem = P_sistem/SNR;

    %%% MEMBUAT KANAL IRF
    beta_k  = 1;
    variance2 = beta_k;
    h_km                                          =
sqrt(variance2/2)*(randn([num_u],[num_m])+((1i)*
randn([num_u],[num_m])));  %small scale fading
berdasar Jumlah antena dan user
    H_irf = h_km;

    %%% MEMBUAT KANAL IRF untuk 2 user
    beta_k_FDB_US2  = 1;
    variance2_FDB_US2 = beta_k_FDB_US2;
    H_irf_us2                                     =
sqrt(variance2_FDB_US2/2)*(randn([total_Rx_US2],
[num_m])+((1i)*randn([total_Rx_US2],[num_m])));
%small scale fading berdasar Jumlah antena dan
user

    %%% MEMBUAT KANAL URLOS
    theta_k1 = -(asind((h_BS-h_us)/d_mk));      %
sudut boresight user terhadap terminal
    beta_k = 1;  % koefisien large scale fading

    fi_urlos     =          -pi     +     (pi-(-
pi).*rand(num_u,1));              % membangkitkan
psi dari -pi sampai pi
    d2_urlos = randi([0 1],num_u,1)>=0.5;
    N2_urlos = 2*d2_urlos-1;
    psi_urlos = N2_urlos.*fi_urlos;
```

```
    lambda_urlos = (3*10^8)/(f*10^9);
    delta_miring_urlos                                  =
lambda_urlos/2*sind(theta_k1);

    for k_urlos = 1:1:num_u
        for m_urlos = 1:1:num_m
            theta_k2_urlos(m_urlos,k_urlos)     =
(((23.5+(lambda_urlos/2)*(k_urlos-1)))-
((lambda_urlos/2)*(m_urlos-
1)))/(500+(delta_miring_urlos*(k_urlos-1))-
(delta_miring_urlos*(m_urlos-1)));
            theta_k_urlos(m_urlos,k_urlos)      =
theta_k1 + (0.01*(k_urlos-1));
            e_kurung_urlos(k_urlos,m_urlos)     =
exp(-i*(m_urlos-
1)*pi*sin(theta_k_urlos(m_urlos,k_urlos)));
        end
        g_km_urlos(k_urlos,:)                           =
(sqrt(beta_k)*exp(i*psi_urlos(k_urlos)))*e_kurun
g_urlos(k_urlos,:);
    end
    H_urlos = g_km_urlos;

    %%% MEMBUAT KANAL urlos untuk 2 user
    theta_k1  =  -(asind((h_BS-h_us)/d_mk));     %
sudut boresight user terhadap terminal user 1
    theta_k2 = -(theta_k1);                          %
sudut boresight user terhadap terminal user 2
    beta_k = 1;          % koefisien large scale
fading

    fi_urlos_us2    =         -pi    +    (pi-(-
pi).*rand(num_u,1));
    d2_urlos_us2 = randi([0 1],num_u,1)>=0.5;
    N2_urlos_us2 = 2*d2_urlos_us2-1;
    psi_urlos_us2 = N2_urlos_us2.*fi_urlos_us2;

    lambda_urlos_us2 = (3*10^8)/(f*10^9);
    delta_miring_urlos_us2                              =
lambda_urlos_us2/2*sind(theta_k1);
```

```
    for k_urlos_us21 = 1:1:num_u
        for m_urlos_us21 = 1:1:num_m

theta_k_urlos_us21(m_urlos_us21,k_urlos_us21)   =
theta_k1 + (0.01*(k_urlos_us21-1));

e_kurung_urlos_us21(k_urlos_us21,m_urlos_us21)  =
exp(-i*(m_urlos_us21-
1)*pi*sin(theta_k_urlos_us21(m_urlos_us21,k_urlo
s_us21)));
        end
        g_km_urlos_us21(k_urlos_us21,:)         =
(sqrt(beta_k)*exp(i*psi_urlos_us2(k_urlos_us21))
)*e_kurung_urlos_us21(k_urlos_us21,:);
    end

    fi_urlos_us22    =         -pi    +    (pi-(-
pi).*rand(num_u,1));              % membangkitkan
psi dari -pi sampai pi
    d2_urlos_us22 = randi([0 1],num_u,1)>=0.5;
    N2_urlos_us22 = 2*d2_urlos_us22-1;
    psi_urlos_us22                              =
N2_urlos_us22.*fi_urlos_us22;

    for k_urlos_us22 = 1:1:num_u
        for m_urlos_us22 = 1:1:num_m

theta_k_urlos_us22(m_urlos_us22,k_urlos_us22)   =
theta_k2 + (0.01*(k_urlos_us22-1));

e_kurung_urlos_us22(k_urlos_us22,m_urlos_us22)  =
exp(-i*(m_urlos_us22-
1)*pi*sin(theta_k_urlos_us22(m_urlos_us22,k_urlo
s_us22)));
        end
        g_km_urlos_us22(k_urlos_us22,:)         =
(sqrt(beta_k)*exp(i*psi_urlos_us22(k_urlos_us22)
))*e_kurung_urlos_us22(k_urlos_us22,:);
    end
```

```
    g_km_urlos_us2                                  =
[g_km_urlos_us21;g_km_urlos_us22];
    H_urlos_us2 = g_km_urlos_us2;

    %%%% MEMBUAT SISTEM FULL-DIGITAL BEAMFORMING
    F1_irf_FDB        = H_irf'*H_irf;
    gamma_irf_FDB                                   =
sqrt(P_sistem/(num_m*NRFt_FDB));
    vrf_irf_FDB       = ones(num_m,NRFt_FDB);
    l1_irf_FDB        = length(num_m);
    nij_irf_FDB       = zeros(num_m,NRFt_FDB);
    vrf_irf1_FDB      = zeros(num_m,NRFt_FDB);
    vrf_new1 = 100;
    tol1 = 1;
    jt1 = 0;

while (tol1 > 1e-5)
    for j1_irf_FDB = 1:1:NRFt_FDB
        vrf_j_FDB = vrf_irf_FDB;
        vrf_j_FDB(:,j1_irf_FDB)= [];
        I1_irf_FDB = eye(NRFt_FDB-1);
        C1_irf_FDB      =      I1_irf_FDB      +
((gamma_irf_FDB^2/variance_sistem)*vrf_j_FDB'*F1
_irf_FDB*vrf_j_FDB);
        G1_irf_FDB                                  =
((gamma_irf_FDB^2/variance_sistem)*F1_irf_FDB)  -
((gamma_irf_FDB^4/variance_sistem^2)*F1_irf_FDB*
vrf_j_FDB*(C1_irf_FDB^(-
1))*vrf_j_FDB'*F1_irf_FDB);
        sigma_irf = 0;
        for i1_irf_FDB = 1:1:num_m
            for L1_irf_FDB = 1:1:num_m
                if L1_irf_FDB ~= i1_irf_FDB
                sigma_irf    =    sigma_irf    +
G1_irf_FDB(i1_irf_FDB,L1_irf_FDB)               *
vrf_irf_FDB(L1_irf_FDB,j1_irf_FDB);
% step 6
                end
            end
```

```
            nij_irf_FDB=sigma_irf;
            if nij_irf_FDB == 0
           vrf_irf_FDB(i1_irf_FDB,j1_irf_FDB)   =
1;
            else

vrf_irf_FDB(i1_irf_FDB,j1_irf_FDB)              =
nij_irf_FDB/abs(nij_irf_FDB);
            end
        end
          cekv_FDB = abs(vrf_irf_FDB);
    end
    I2_irf_FDB   = eye(NRFr_FDB);

VdVdh_irf_FDB=(sqrt(gamma_irf_FDB^2))*I2_irf_FDB
;
    Heff_irf_FDB = H_irf*vrf_irf_FDB;
    Q_irf_FDB = vrf_irf_FDB'*vrf_irf_FDB;
    svd_irf_FDB    =    Heff_irf_FDB*(Q_irf_FDB^(-
0.5));

[U_irf_FDB,S_irf_FDB,V_irf_FDB]=svd(svd_irf_FDB)
;
    Ue_irf_FDB = V_irf_FDB(1:NRFt_FDB,1:Ns_FDB);
    Te_irf_FDB                                    =
sqrt(P_sistem/NRFt_FDB)*(eye(Ns_FDB));
    Vd_irf_FDB          =          (Q_irf_FDB^(-
0.5))*Ue_irf_FDB*Te_irf_FDB;

    vrf_old1 = vrf_new1;
    vrf_new1                                      =
trace(vrf_irf_FDB*Vd_irf_FDB*Vd_irf_FDB'*vrf_irf
_FDB');
    tol1 = vrf_new1 - vrf_old1;

    jt1 = jt1 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO
```

```
Vt_irf_FDB = vrf_irf_FDB*Vd_irf_FDB;
F2_irf_FDB = H_irf*Vt_irf_FDB*Vt_irf_FDB'*H_irf';

wrf_irf_FDB = ones(num_u,NRFr_FDB);
l2_irf_FDB = length(num_u);
nij2_irf_FDB = zeros(num_u,NRFr_FDB);
Wrf_irf1_FDB = zeros(num_u,NRFr_FDB);
vrf_new2 = 100;
tol2 = 1;
jt2 = 0;
while (tol2 > 1e-5)
    for                j2_irf_FDB                =
1:1:NRFr_FDB                                      %step
2
        wrf_j_FDB            = wrf_irf_FDB;
        wrf_j_FDB(:,j2_irf_FDB)       = [];
        I2_irf_FDB                 = eye(NRFr_FDB-
1);   %step 3&4
        C2_irf_FDB                  = I2_irf_FDB +
(((1/num_u)/variance_sistem)*wrf_j_FDB'*F2_irf_F
DB*wrf_j_FDB);
        G2_irf_FDB                                 =
(((1/num_u)/variance_sistem)*F2_irf_FDB)          -
(((1/num_u)^2/variance_sistem^2)*F2_irf_FDB*wrf_
j_FDB*(C2_irf_FDB^(-1))*wrf_j_FDB'*F2_irf_FDB);
        sigma2_FDB = 0;
        for i2_irf_FDB = 1:1:num_u
            %sigma2_FDB = 0;
            for L2_irf_FDB = 1:1:num_u
                %sigma2_FDB = 0;
                if L2_irf_FDB ~= i2_irf_FDB
                sigma2_FDB    =    sigma2_FDB    +
G2_irf_FDB(i2_irf_FDB,L2_irf_FDB)                 *
wrf_irf_FDB(L2_irf_FDB,j2_irf_FDB);
                end
            end
            nij2_irf_FDB=sigma2_FDB;
            if nij2_irf_FDB == 0
            wrf_irf_FDB(i2_irf_FDB,j2_irf_FDB)  =
```

```
1;
            else

wrf_irf_FDB(i2_irf_FDB,j2_irf_FDB)              =
nij2_irf_FDB/abs(nij2_irf_FDB);
            end
        end
    end
    j_IRF_FDB=
wrf_irf_FDB'*H_irf*vrf_irf_FDB*Vd_irf_FDB*Vd_irf
_FDB'*vrf_irf_FDB'*H_irf'*wrf_irf_FDB           +
variance_sistem*wrf_irf_FDB'*wrf_irf_FDB;
    J_IRF_FDB           = inv(j_IRF_FDB);
    Wd_irf_FDB                                  =
J_IRF_FDB*wrf_irf_FDB'*H_irf*vrf_irf_FDB*Vd_irf_
FDB;
    Wt_irf_FDB      = wrf_irf_FDB*Wd_irf_FDB;

    vrf_old2 = vrf_new2;
    vrf_new2                                    =
trace(wrf_irf_FDB*Wd_irf_FDB*Wd_irf_FDB'*wrf_irf
_FDB');
    tol2 = vrf_new2 - vrf_old2;

jt2 = jt2 + 1;
end

SE_IRF_FDB(SNRLoop,n_sistem)                    =
real(log2(det((eye(num_u))                      +
(1/variance_sistem)*Wt_irf_FDB*(inv(Wt_irf_FDB'*
Wt_irf_FDB))*Wt_irf_FDB'*H_irf*Vt_irf_FDB*Vt_irf
_FDB'*H_irf')));

%%%%%% MEMBUAT SISTEM HYBRID BEAMFORMING
F1_irf_HB = H_irf'*H_irf;
gamma_irf_HB = sqrt(P_sistem/(num_m*NRFt_HB));
vrf_irf_HB = ones(num_m,NRFt_HB);
l1_irf_HB = length(num_m);
nij_irf_HB = zeros(num_m,NRFt_HB);
vrf_irf1_HB = zeros(num_m,NRFt_HB);
```

```
vrf_new3 = 100;
tol3 = 1;
jt3 = 0;

while (tol3 > 1e-5)
    for j1_irf_HB = 1:1:NRFt_HB
        vrf_j_HB = vrf_irf_HB;
        vrf_j_HB(:,j1_irf_HB) = [];
        I1_irf_HB = eye(NRFt_HB-1);
        C1_irf_HB                    = I1_irf_HB +
((gamma_irf_HB^2/variance_sistem)*vrf_j_HB'*F1_i
rf_HB*vrf_j_HB);
        G1_irf_HB                                  =
((gamma_irf_HB^2/variance_sistem)*F1_irf_HB)    -
((gamma_irf_HB^4/variance_sistem^2)*F1_irf_HB*vr
f_j_HB*(C1_irf_HB^(-1))*vrf_j_HB'*F1_irf_HB);
        sigma_HB = 0;
        for i1_irf_HB = 1:1:num_m
            for L1_irf_HB = 1:1:num_m
                if L1_irf_HB ~= i1_irf_HB
                sigma_HB     =     sigma_HB     +
G1_irf_HB(i1_irf_HB,L1_irf_HB)                     *
vrf_irf_HB(L1_irf_HB,j1_irf_HB);
                end
            end
            nij_irf_HB=sigma_HB;
            if nij_irf_HB == 0
            vrf_irf_HB(i1_irf_HB,j1_irf_HB) = 1;
            else
               vrf_irf_HB(i1_irf_HB,j1_irf_HB)  =
nij_irf_HB/abs(nij_irf_HB);
            end
        end
          cekv_FDB = abs(vrf_irf_HB);
    end
    I2_irf_HB                = eye(NRFr_HB);
    VdVdh_irf_HB                                   =
(sqrt(gamma_irf_HB^2))*I2_irf_HB;
    Heff_irf_HB    = H_irf*vrf_irf_HB;
    Q_irf_HB                                       =
```

```
vrf_irf_HB'*vrf_irf_HB;
    svd_irf_HB                                    =
Heff_irf_HB*(Q_irf_HB^(-0.5));
   [U_irf_HB,S_irf_HB,V_irf_HB]=svd(svd_irf_HB);
    Ue_irf_HB = V_irf_HB(1:NRFt_HB,1:Ns_HB);
    Te_irf_HB                                     =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_irf_HB           =           (Q_irf_HB^(-
0.5))*Ue_irf_HB*Te_irf_HB;

    vrf_old3 = vrf_new3;
    vrf_new3                                      =
trace(vrf_irf_HB*Vd_irf_HB*Vd_irf_HB'*vrf_irf_HB
');
    tol3 = vrf_new3 - vrf_old3;

    jt3 = jt3 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_HB = vrf_irf_HB*Vd_irf_HB;
F2_irf_HB = H_irf*Vt_irf_HB*Vt_irf_HB'*H_irf';
wrf_irf_HB       = ones(num_u,NRFr_HB);
l2_irf_HB        = length(num_u);
nij2_irf_HB      = zeros(num_u,NRFr_HB);
Wrf_irf1_HB      = zeros(num_u,NRFr_HB);
vrf_new4 = 100;
tol4 = 1;
jt4 = 0;

while (tol4 > 1e-5)
    for j2_irf_HB = 1:1:NRFr_HB
        wrf_j_HB  = wrf_irf_HB;
        wrf_j_HB(:,j2_irf_HB)      = [];
        I2_irf_HB = eye(NRFr_HB-1);
        C2_irf_HB       =       I2_irf_HB       +
(((1/num_u)/variance_sistem)*wrf_j_HB'*F2_irf_HB
*wrf_j_HB);
```

```
        G2_irf_HB                                          =
(((1/num_u)/variance_sistem)*F2_irf_HB)                    -
(((1/num_u)^2/variance_sistem^2)*F2_irf_HB*wrf_j
_HB*(C2_irf_HB^(-1))*wrf_j_HB'*F2_irf_HB);
        sigma2_HB = 0;
        for i2_irf_HB = 1:1:num_u
            for L2_irf_HB = 1:1:num_u
                if L2_irf_HB ~= i2_irf_HB
                sigma2_HB    =    sigma2_HB    +
G2_irf_HB(i2_irf_HB,L2_irf_HB)                             *
wrf_irf_HB(L2_irf_HB,j2_irf_HB);
                end
            end
            nij2_irf_HB=sigma2_HB;
            if nij2_irf_HB == 0
            wrf_irf_HB(i2_irf_HB,j2_irf_HB) = 1;
            else
               wrf_irf_HB(i2_irf_HB,j2_irf_HB)  =
nij2_irf_HB/abs(nij2_irf_HB);
            end
        end
    end
    j_IRF_HB                                               =
wrf_irf_HB'*H_irf*vrf_irf_HB*Vd_irf_HB*Vd_irf_HB
'*vrf_irf_HB'*H_irf'*wrf_irf_HB                            +
variance_sistem*wrf_irf_HB'*wrf_irf_HB;
    J_IRF_HB           = inv(j_IRF_HB);
    Wd_irf_HB                                              =
J_IRF_HB*wrf_irf_HB'*H_irf*vrf_irf_HB*Vd_irf_HB;
    Wt_irf_HB      = wrf_irf_HB*Wd_irf_HB;

    vrf_old4 = vrf_new4;
    vrf_new4                                               =
trace(wrf_irf_HB*Wd_irf_HB*Wd_irf_HB'*wrf_irf_HB
');
    tol4 = vrf_new4 - vrf_old4;

jt4 = jt4 + 1;
end
```

```
SE_IRF_HB(SNRLoop,n_sistem)                          =
real(log2(det((eye(num_u))                           +
(1/variance_sistem)*Wt_irf_HB*(inv(Wt_irf_HB'*Wt
_irf_HB))*Wt_irf_HB'*H_irf*Vt_irf_HB*Vt_irf_HB'*
H_irf')));

%%%% MEMBUAT SISTEM FULL-DIGITAL BEAMFORMING

F1_urlos_FDB         = H_urlos'*H_urlos;
gamma_urlos_FDB                                      =
sqrt(P_sistem/(num_m*NRFt_FDB));
vrf_urlos_FDB        = ones(num_m,NRFt_FDB);
l1_urlos_FDB         = length(num_m);
nij_urlos_FDB        = zeros(num_m,NRFt_FDB);
vrf_urlos1_FDB       = zeros(num_m,NRFt_FDB);
vrf_new5 = 100;
tol5 = 1;
jt5 = 0;

while (tol5 > 1e-5)
    for j1_urlos_FDB = 1:1:NRFt_FDB
        vrfj_urlos_FDB                               =
vrf_urlos_FDB;
        vrfj_urlos_FDB(:,j1_urlos_FDB)       = [];
        I1_urlos_FDB               = eye(NRFt_FDB-
1);
        C1_urlos_FDB               = I1_urlos_FDB
+
((gamma_urlos_FDB^2/variance_sistem)*vrfj_urlos_
FDB'*F1_urlos_FDB*vrfj_urlos_FDB);
        G1_urlos_FDB                                 =
((gamma_urlos_FDB^2/variance_sistem)*F1_urlos_FD
B)                                                   -
((gamma_urlos_FDB^4/variance_sistem^2)*F1_urlos_
FDB*vrfj_urlos_FDB*(C1_urlos_FDB^(-
1))*vrfj_urlos_FDB'*F1_urlos_FDB);
        sigma_urlos_FDB = 0;
        for i1_urlos_FDB = 1:1:num_m
            for L1_urlos_FDB = 1:1:num_m
                if L1_urlos_FDB ~= i1_urlos_FDB
```

```
                sigma_urlos_FDB = sigma_urlos_FDB
+    G1_urlos_FDB(i1_urlos_FDB,L1_urlos_FDB)    *
vrf_urlos_FDB(L1_urlos_FDB,j1_urlos_FDB);
                end
            end
            nij_urlos_FDB=sigma_urlos_FDB;
            if nij_urlos_FDB == 0

vrf_urlos_FDB(i1_urlos_FDB,j1_urlos_FDB) = 1;
            else

vrf_urlos_FDB(i1_urlos_FDB,j1_urlos_FDB)        =
nij_urlos_FDB/abs(nij_urlos_FDB);
            end
        end
          cekv_FDB = abs(vrf_urlos_FDB);
    end
    I2_urlos_FDB = eye(NRFr_FDB);
    VdVdh_urlos_FDB                             =
(sqrt(gamma_urlos_FDB^2))*I2_urlos_FDB;
    Heff_urlos_FDB = H_urlos*vrf_urlos_FDB;
    Q_urlos_FDB = vrf_urlos_FDB'*vrf_urlos_FDB;
    svd_urlos_FDB                               =
Heff_urlos_FDB*(Q_urlos_FDB^(-0.5));
    [U_urlos_FDB,S_urlos_FDB,V_urlos_FDB]       =
svd(svd_urlos_FDB);
    Ue_urlos_FDB                                =
V_urlos_FDB(1:NRFt_FDB,1:Ns_FDB);
    Te_urlos_FDB                                =
sqrt(P_sistem/NRFt_FDB)*(eye(Ns_FDB));
    Vd_urlos_FDB        =        (Q_urlos_FDB^(-
0.5))*Ue_urlos_FDB*Te_urlos_FDB;

    vrf_old5 = vrf_new5;
    vrf_new5                                    =
trace(vrf_urlos_FDB*Vd_urlos_FDB*Vd_urlos_FDB'*v
rf_urlos_FDB');
    tol5 = vrf_new5 - vrf_old5;

    jt5 = jt5 + 1;
```

```
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_FDB                                    =
vrf_urlos_FDB*Vd_urlos_FDB;
F2_urlos_FDB                                    =
H_urlos*Vt_urlos_FDB*Vt_urlos_FDB'*H_urlos';

wrf_urlos_FDB       = ones(num_u,NRFr_FDB);
l2_urlos_FDB        = length(num_u);
nij2_urlos_FDB      = zeros(num_u,NRFr_FDB);
Wrf_urlos1_FDB      = zeros(num_u,NRFr_FDB);
vrf_new6 = 100;
tol6 = 1;
jt6 = 0;

while (tol6 > 1e-5)
    for j2_irf_FDB = 1:1:NRFr_FDB
        wrfj_urlos_FDB = wrf_urlos_FDB;
        wrfj_urlos_FDB(:,j2_irf_FDB)      = [];
        I2_urlos_FDB = eye(NRFr_FDB-1);
        C2_urlos_FDB=       I2_urlos_FDB       +
(((1/num_u)/variance_sistem)*wrfj_urlos_FDB'*F2_
urlos_FDB*wrfj_urlos_FDB);
        G2_urlos_FDB                            =
(((1/num_u)/variance_sistem)*F2_urlos_FDB)      -
(((1/num_u)^2/variance_sistem^2)*F2_urlos_FDB*wr
fj_urlos_FDB*(C2_urlos_FDB^(-
1))*wrfj_urlos_FDB'*F2_urlos_FDB);
        sigma2_urlos_FDB = 0;
        for i2_urlos_FDB = 1:1:num_u
            for L2_urlos_FDB = 1:1:num_u
                if L2_urlos_FDB ~= i2_urlos_FDB
                sigma2_urlos_FDB                =
sigma2_urlos_FDB                                +
G2_urlos_FDB(i2_urlos_FDB,L2_urlos_FDB)         *
wrf_urlos_FDB(L2_urlos_FDB,j2_irf_FDB);
                end
```

```
            end
            nij2_urlos_FDB=sigma2_urlos_FDB;
            if nij2_urlos_FDB == 0

wrf_urlos_FDB(i2_urlos_FDB,j2_irf_FDB) = 1;
            else

wrf_urlos_FDB(i2_urlos_FDB,j2_irf_FDB)          =
nij2_urlos_FDB/abs(nij2_urlos_FDB);
            end
        end
    end
    j_urlos_FDB                                 =
wrf_urlos_FDB'*H_urlos*vrf_urlos_FDB*Vd_urlos_FD
B*Vd_urlos_FDB'*vrf_urlos_FDB'*H_urlos'*wrf_urlo
s_FDB                                           +
variance_sistem*wrf_urlos_FDB'*wrf_urlos_FDB;
    J_urlos_FDB  = inv(j_urlos_FDB);
    Wd_urlos_FDB                                =
J_urlos_FDB*wrf_urlos_FDB'*H_urlos*vrf_urlos_FDB
*Vd_urlos_FDB;
    Wt_urlos_FDB = wrf_urlos_FDB*Wd_urlos_FDB;

    vrf_old6 = vrf_new6;
    vrf_new6                                    =
trace(wrf_urlos_FDB*Wd_urlos_FDB*Wd_urlos_FDB'*w
rf_urlos_FDB');
    tol6 = vrf_new6 - vrf_old6;

jt6 = jt6 + 1;
end

SE_URLOS_FDB(SNRLoop,n_sistem)                  =
real(log2(det((eye(num_u))                      +
(1/variance_sistem)*Wt_urlos_FDB*(inv(Wt_urlos_F
DB'*Wt_urlos_FDB))*Wt_urlos_FDB'*H_urlos*Vt_urlo
s_FDB*Vt_urlos_FDB'*H_urlos')));

%%%%%% MEMBUAT SISTEM HYBRID BEAMFORMING
F1_urlos_HB        = H_urlos'*H_urlos;
```

```
gamma_urlos_HB                                  =
sqrt(P_sistem/(num_m*NRFt_HB));
vrf_urlos_HB        = ones(num_m,NRFt_HB);
l1_urlos_HB         = length(num_m);
nij_urlos_HB        = zeros(num_m,NRFt_HB);
vrf_urlos1_HB       = zeros(num_m,NRFt_HB);
vrf_new7          = 100;
tol7              = 1;
jt7               = 0;

while (tol7 > 1e-5)
    for j1_urlos_HB = 1:1:NRFt_HB
        vrfj_urlos_HB   = vrf_urlos_HB;
        vrfj_urlos_HB(:,j1_urlos_HB) = [];
        I1_urlos_HB    = eye(NRFt_HB-1);
        C1_urlos_HB      =      I1_urlos_HB      +
((gamma_urlos_HB^2/variance_sistem)*vrfj_urlos_H
B'*F1_urlos_HB*vrfj_urlos_HB);
        G1_urlos_HB                             =
((gamma_urlos_HB^2/variance_sistem)*F1_urlos_HB)
-
((gamma_urlos_HB^4/variance_sistem^2)*F1_urlos_H
B*vrfj_urlos_HB*(C1_urlos_HB^(-
1))*vrfj_urlos_HB'*F1_urlos_HB);
        sigma_urlos_HB = 0;
        for i1_urlos_HB = 1:1:num_m
            for L1_urlos_HB = 1:1:num_m
                if L1_urlos_HB ~= i1_urlos_HB
                sigma_urlos_HB = sigma_urlos_HB +
G1_urlos_HB(i1_urlos_HB,L1_urlos_HB)            *
vrf_urlos_HB(L1_urlos_HB,j1_urlos_HB);
                end
            end
            nij_urlos_HB=sigma_urlos_HB;
            if nij_urlos_HB == 0

vrf_urlos_HB(i1_urlos_HB,j1_urlos_HB) = 1;
            else

vrf_urlos_HB(i1_urlos_HB,j1_urlos_HB)           =
```

```
nij_urlos_HB/abs(nij_urlos_HB);
            end
        end
          cekv_FDB = abs(vrf_urlos_HB);
    end
    I2_urlos_HB  = eye(NRFr_HB);
    VdVdh_urlos_HB                                  =
(sqrt(gamma_urlos_HB^2))*I2_urlos_HB;
    Heff_urlos_HB = H_urlos*vrf_urlos_HB;
    Q_urlos_HB = vrf_urlos_HB'*vrf_urlos_HB;
    svd_urlos_HB  =  Heff_urlos_HB*(Q_urlos_HB^(-
0.5));
    [U_urlos_HB,S_urlos_HB,V_urlos_HB]          =
svd(svd_urlos_HB);
    Ue_urlos_HB = V_urlos_HB(1:NRFt_HB,1:Ns_HB);
    Te_urlos_HB                                     =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_urlos_HB         =         (Q_urlos_HB^(-
0.5))*Ue_urlos_HB*Te_urlos_HB;

    vrf_old7 = vrf_new7;
    vrf_new7                                        =
trace(vrf_urlos_HB*Vd_urlos_HB*Vd_urlos_HB'*vrf_
urlos_HB');
    tol7 = vrf_new7 - vrf_old7;

    jt7 = jt7 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_HB = vrf_urlos_HB*Vd_urlos_HB;
F2_urlos_HB                                     =
H_urlos*Vt_urlos_HB*Vt_urlos_HB'*H_urlos';

wrf_urlos_HB       = ones(num_u,NRFr_HB);
l2_urlos_HB        = length(num_u);
nij2_urlos_HB      = zeros(num_u,NRFr_HB);
Wrf_urlos1_HB      = zeros(num_u,NRFr_HB);
```

```
vrf_new8 = 100;
tol8 = 1;
jt8 = 0;

while (tol8 > 1e-5)
    for j2_urlos_HB = 1:1:NRFr_HB
        wrfj_urlos_HB= wrf_urlos_HB;
        wrfj_urlos_HB(:,j2_urlos_HB)      = [];
        I2_urlos_HB = eye(NRFr_HB-1);
        C2_urlos_HB     =     I2_urlos_HB     +
(((1/num_u)/variance_sistem)*wrfj_urlos_HB'*F2_u
rlos_HB*wrfj_urlos_HB);
        G2_urlos_HB                                =
(((1/num_u)/variance_sistem)*F2_urlos_HB)       -
(((1/num_u)^2/variance_sistem^2)*F2_urlos_HB*wrf
j_urlos_HB*(C2_urlos_HB^(-
1))*wrfj_urlos_HB'*F2_urlos_HB);
        sigma2_urlos_HB = 0;
        for i2_irf_HB = 1:1:num_u
            for L2_irf_HB = 1:1:num_u
                if L2_irf_HB ~= i2_irf_HB
                sigma2_urlos_HB = sigma2_urlos_HB
+       G2_urlos_HB(i2_irf_HB,L2_irf_HB)       *
wrf_urlos_HB(L2_irf_HB,j2_urlos_HB);
                end
            end
            nij2_urlos_HB=sigma2_urlos_HB;
            if nij2_urlos_HB == 0

wrf_urlos_HB(i2_irf_HB,j2_urlos_HB) = 1;
            else

wrf_urlos_HB(i2_irf_HB,j2_urlos_HB)             =
nij2_urlos_HB/abs(nij2_urlos_HB);
            end
        end
    end
    j_urlos_HB                                 =
wrf_urlos_HB'*H_urlos*vrf_urlos_HB*Vd_urlos_HB*V
d_urlos_HB'*vrf_urlos_HB'*H_urlos'*wrf_urlos_HB
```

```
+ variance_sistem*wrf_urlos_HB'*wrf_urlos_HB;
    J_urlos_HB  = inv(j_urlos_HB);
    Wd_urlos_HB                                    =
J_urlos_HB*wrf_urlos_HB'*H_urlos*vrf_urlos_HB*Vd
_urlos_HB;
    Wt_urlos_HB  = wrf_urlos_HB*Wd_urlos_HB;

    vrf_old8 = vrf_new8;
    vrf_new8                                       =
trace(wrf_urlos_HB*Wd_urlos_HB*Wd_urlos_HB'*wrf_
urlos_HB');
    tol8 = vrf_new8 - vrf_old8;

jt8 = jt8 + 1;
end

SE_URLOS_HB(SNRLoop,n_sistem)                      =
real(log2(det((eye(num_u))                         +
(1/variance_sistem)*Wt_urlos_HB*(inv(Wt_urlos_HB
'*Wt_urlos_HB))*Wt_urlos_HB'*H_urlos*Vt_urlos_HB
*Vt_urlos_HB'*H_urlos')));

%%%%% MEMBUAT SISTEM DENGAN ANTENA TX=64

F1_irf_TX64         = H_irf'*H_irf;
gamma_irf_TX64                                     =
sqrt(P_sistem/(num_m_TX64*NRFt_HB));
vrf_irf_TX64        = ones(num_m_TX64,NRFt_HB);
l1_irf_TX64         = length(num_m_TX64);
nij_irf_TX64        = zeros(num_m_TX64,NRFt_HB);
vrf_irf1_TX64       = zeros(num_m_TX64,NRFt_HB);
vrf_new9          = 100;
tol9              = 1;
jt9               = 0;

while (tol9 > 1e-5)
    for j1_irf_TX64 = 1:1:NRFt_HB
        vrfj_irf_TX64  = vrf_irf_TX64;
        vrfj_irf_TX64(:,j1_irf_TX64) = [];
        I1_irf_TX64 = eye(NRFt_HB-1);
```

```
        C1_irf_TX64      =      I1_irf_TX64      +
((gamma_irf_TX64^2/variance_sistem)*vrfj_irf_TX6
4'*F1_irf_TX64*vrfj_irf_TX64);
        G1_irf_TX64                                =
((gamma_irf_TX64^2/variance_sistem)*F1_irf_TX64)
-
((gamma_irf_TX64^4/variance_sistem^2)*F1_irf_TX6
4*vrfj_irf_TX64*(C1_irf_TX64^(-
1))*vrfj_irf_TX64'*F1_irf_TX64);
        sigma_irf_TX64 = 0;
        for i1_irf_TX64 = 1:1:num_m_TX64
            for L1_irf_TX64 = 1:1:num_m_TX64
                if L1_irf_TX64 ~= i1_irf_TX64
                sigma_irf_TX64 = sigma_irf_TX64 +
G1_irf_TX64(i1_irf_TX64,L1_irf_TX64)             *
vrf_irf_TX64(L1_irf_TX64,j1_irf_TX64);
                end
            end
            nij_irf_TX64=sigma_irf_TX64;
            if nij_irf_TX64 == 0

vrf_irf_TX64(i1_irf_TX64,j1_irf_TX64) = 1;
            else

vrf_irf_TX64(i1_irf_TX64,j1_irf_TX64)           =
nij_irf_TX64/abs(nij_irf_TX64);
            end
        end
          cekv_FDB = abs(vrf_irf_TX64);
    end
    I2_irf_TX64  = eye(NRFr_HB);
    VdVdh_irf_TX64                              =
(sqrt(gamma_irf_TX64^2))*I2_irf_TX64;
    Heff_irf_TX64 = H_irf*vrf_irf_TX64;
    Q_irf_TX64 = vrf_irf_TX64'*vrf_irf_TX64;
    svd_irf_TX64  =  Heff_irf_TX64*(Q_irf_TX64^(-
0.5));
    [U_irf_TX64,S_irf_TX64,V_irf_TX64]          =
svd(svd_irf_TX64);
    Ue_irf_TX64 = V_irf_TX64(1:NRFt_HB,1:Ns_HB);
```

```
    Te_irf_TX64                                          =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_irf_TX64                    = (Q_irf_TX64^(-
0.5))*Ue_irf_TX64*Te_irf_TX64;

    vrf_old9 = vrf_new9;
    vrf_new9                                             =
trace(vrf_irf_TX64*Vd_irf_TX64*Vd_irf_TX64'*vrf_
irf_TX64');
    tol9 = vrf_new9 - vrf_old9;

    jt9 = jt9 + 9;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_TX64  = vrf_irf_TX64*Vd_irf_TX64;
F2_irf_TX64=
H_irf*Vt_irf_TX64*Vt_irf_TX64'*H_irf';

wrf_irf_TX64        = ones(num_u_TX64,NRFr_HB);
l2_irf_TX64         = length(num_u_TX64);
nij2_irf_TX64       = zeros(num_u_TX64,NRFr_HB);
Wrf_irf1_TX64       = zeros(num_u_TX64,NRFr_HB);
vrf_new10 = 100;
tol10 = 1;
jt10 = 0;

while (tol10 > 1e-5)
    for j2_irf_TX64 = 1:1:NRFr_HB
        wrfj_IRF_TX64  = wrf_irf_TX64;
        wrfj_IRF_TX64(:,j2_irf_TX64)= [];
        I2_irf_TX64 = eye(NRFr_HB-1);
        C2_irf_TX64      =      I2_irf_TX64      +
(((1/num_u_TX64)/variance_sistem)*wrfj_IRF_TX64'
*F2_irf_TX64*wrfj_IRF_TX64);
        G2_irf_TX64                                  =
(((1/num_u_TX64)/variance_sistem)*F2_irf_TX64)  -
(((1/num_u_TX64)^2/variance_sistem^2)*F2_irf_TX6
```

```
4*wrfj_IRF_TX64*(C2_irf_TX64^(-
1))*wrfj_IRF_TX64'*F2_irf_TX64);
        sigma2_IRF_TX64 = 0;
        for i2_irf_TX64 = 1:1:num_u_TX64
            for L2_irf_TX64 = 1:1:num_u_TX64
                if L2_irf_TX64 ~= i2_irf_TX64
                sigma2_IRF_TX64 = sigma2_IRF_TX64
+      G2_irf_TX64(i2_irf_TX64,L2_irf_TX64)      *
wrf_irf_TX64(L2_irf_TX64,j2_irf_TX64);
                end
            end
            nij2_irf_TX64=sigma2_IRF_TX64;
            if nij2_irf_TX64 == 0

wrf_irf_TX64(i2_irf_TX64,j2_irf_TX64) = 1;
            else

wrf_irf_TX64(i2_irf_TX64,j2_irf_TX64)           =
nij2_irf_TX64/abs(nij2_irf_TX64);
            end
        end
    end
    j_IRF_TX64                                  =
wrf_irf_TX64'*H_irf*vrf_irf_TX64*Vd_irf_TX64*Vd_
irf_TX64'*vrf_irf_TX64'*H_irf'*wrf_irf_TX64     +
variance_sistem*wrf_irf_TX64'*wrf_irf_TX64;
    J_IRF_TX64           = inv(j_IRF_TX64);
    Wd_irf_TX64                                 =
J_IRF_TX64*wrf_irf_TX64'*H_irf*vrf_irf_TX64*Vd_i
rf_TX64;
    Wt_irf_TX64      = wrf_irf_TX64*Wd_irf_TX64;

    vrf_old10 = vrf_new10;
    vrf_new10                                   =
trace(wrf_irf_TX64*Wd_irf_TX64*Wd_irf_TX64'*wrf_
irf_TX64');
    tol10 = vrf_new10 - vrf_old10;

jt10 = jt10 + 1;
end
```

```
SE_IRF_TX64(SNRLoop,n_sistem)                    =
real(log2(det((eye(num_u_TX64))                  +
(1/variance_sistem)*Wt_irf_TX64*(inv(Wt_irf_TX64
'*Wt_irf_TX64))*Wt_irf_TX64'*H_irf*Vt_irf_TX64*V
t_irf_TX64'*H_irf')));

%%%%%  MEMBUAT SISTEM DENGAN ANTENA TX=32

H_irf_TX32 =  H_irf(:,1:num_m_TX32);
F1_irf_TX32                                      =
H_irf_TX32(:,1:num_m_TX32)'*H_irf_TX32(:,1:num_m
_TX32);
gamma_irf_TX32                                   =
sqrt(P_sistem/(num_m_TX32*NRFt_HB));
vrf_irf_TX32                                     =
ones(num_m_TX32,NRFt_HB);                % step 1
l1_irf_TX32        = length(num_m_TX32);
nij_irf_TX32       = zeros(num_m_TX32,NRFt_HB);
vrf_irf1_TX32      = zeros(num_m_TX32,NRFt_HB);
vrf_new11         = 100;
tol11             = 1;
jt11              = 0;

while (tol11 > 1e-5)
    for j1_irf_TX32 = 1:1:NRFt_HB
        vrfj_IRF_TX32             = vrf_irf_TX32;
        vrfj_IRF_TX32(:,j1_irf_TX32)      = [];
        I1_irf_TX32                = eye(NRFt_HB-
1);
        C1_irf_TX32               = I1_irf_TX32 +
((gamma_irf_TX32^2/variance_sistem)*vrfj_IRF_TX3
2'*F1_irf_TX32*vrfj_IRF_TX32);           % step 3
        G1_irf_TX32                              =
((gamma_irf_TX32^2/variance_sistem)*F1_irf_TX32)
-
((gamma_irf_TX32^4/variance_sistem^2)*F1_irf_TX3
2*vrfj_IRF_TX32*(C1_irf_TX32^(-
1))*vrfj_IRF_TX32'*F1_irf_TX32);
        sigma_IRF_TX32 = 0;
```

```
        for i1_irf_TX32 = 1:1:num_m_TX32
            for L1_irf_TX32 = 1:1:num_m_TX32
                if L1_irf_TX32 ~= i1_irf_TX32
                sigma_IRF_TX32 = sigma_IRF_TX32 +
G1_irf_TX32(i1_irf_TX32,L1_irf_TX32)            *
vrf_irf_TX32(L1_irf_TX32,j1_irf_TX32);
                end
            end
            nij_irf_TX32=sigma_IRF_TX32;
            if nij_irf_TX32 == 0

vrf_irf_TX32(i1_irf_TX32,j1_irf_TX32) = 1;
            else

vrf_irf_TX32(i1_irf_TX32,j1_irf_TX32)           =
nij_irf_TX32/abs(nij_irf_TX32);
            end
        end
          cekv_FDB = abs(vrf_irf_TX32);
    end
    I2_irf_TX32 = eye(NRFr_HB);
    VdVdH_irf_32_TX32                           =
(sqrt(gamma_irf_TX32^2))*I2_irf_TX32;
    Heff_irf_TX32 = H_irf_TX32*vrf_irf_TX32;
    Q_irf_TX32 = vrf_irf_TX32'*vrf_irf_TX32;
    svd_irf_TX32  =  Heff_irf_TX32*(Q_irf_TX32^(-
0.5));
    [U_irf_TX32,S_irf_TX32,V_irf_TX32]          =
svd(svd_irf_TX32);
    Ue_irf_TX32 = V_irf_TX32(1:NRFt_HB,1:Ns_HB);
    Te_irf_TX32                                 =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_irf_TX32                 = (Q_irf_TX32^(-
0.5))*Ue_irf_TX32*Te_irf_TX32;

    vrf_old11 = vrf_new11;
    vrf_new11                                   =
trace(vrf_irf_TX32*Vd_irf_TX32*Vd_irf_TX32'*vrf_
irf_TX32');
    tol11 = vrf_new11 - vrf_old11;
```

```
    jt11 = jt11 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_TX32                                          =
vrf_irf_TX32*Vd_irf_TX32;
F2_irf_TX32                                          =
H_irf_TX32*Vt_irf_TX32*Vt_irf_TX32'*H_irf_TX32';

wrf_irf_TX32        = ones(num_u_TX32,NRFr_HB);
12_irf_TX32         = length(num_u_TX32);
nij2_irf_TX32       = zeros(num_u_TX32,NRFr_HB);
Wrf_irf1_TX32       = zeros(num_u_TX32,NRFr_HB);
vrf_new12 = 100;
tol12 = 1;
jt12 = 0;

while (tol12 > 1e-5)
    for j2_irf_TX32 = 1:1:NRFr_HB
        wrfj_IRF_TX32           = wrf_irf_TX32;
        wrfj_IRF_TX32(:,j2_irf_TX32)      = [];
        I2_irf_TX32                = eye(NRFr_HB-
1);
        C2_irf_TX32               = I2_irf_TX32 +
(((1/num_u_TX32)/variance_sistem)*wrfj_IRF_TX32'
*F2_irf_TX32*wrfj_IRF_TX32);
        G2_irf_TX32                                  =
(((1/num_u_TX32)/variance_sistem)*F2_irf_TX32)  -
(((1/num_u_TX32)^2/variance_sistem^2)*F2_irf_TX3
2*wrfj_IRF_TX32*(C2_irf_TX32^(-
1))*wrfj_IRF_TX32'*F2_irf_TX32);
        sigma2_IRF_TX32 = 0;
        for i2_irf_TX32 = 1:1:num_u_TX32
            for L2_irf_TX32 = 1:1:num_u_TX32
                if L2_irf_TX32 ~= i2_irf_TX32
                sigma2_IRF_TX32 = sigma2_IRF_TX32
+     G2_irf_TX32(i2_irf_TX32,L2_irf_TX32)     *
```

```
wrf_irf_TX32(L2_irf_TX32,j2_irf_TX32);
                end
            end
            nij2_irf_TX32=sigma2_IRF_TX32;
            if nij2_irf_TX32 == 0

wrf_irf_TX32(i2_irf_TX32,j2_irf_TX32) = 1;
            else

wrf_irf_TX32(i2_irf_TX32,j2_irf_TX32)           =
nij2_irf_TX32/abs(nij2_irf_TX32);
            end
        end
    end
    j_IRF_TX32                                  =
wrf_irf_TX32'*H_irf_TX32*vrf_irf_TX32*Vd_irf_TX3
2*Vd_irf_TX32'*vrf_irf_TX32'*H_irf_TX32'*wrf_irf
_TX32                                           +
variance_sistem*wrf_irf_TX32'*wrf_irf_TX32;
    J_IRF_TX32          = inv(j_IRF_TX32);
    Wd_irf_TX32                                 =
J_IRF_TX32*wrf_irf_TX32'*H_irf_TX32*vrf_irf_TX32
*Vd_irf_TX32;
    Wt_irf_TX32      = wrf_irf_TX32*Wd_irf_TX32;

    vrf_old12 = vrf_new12;
    vrf_new12                                   =
trace(wrf_irf_TX32*Wd_irf_TX32*Wd_irf_TX32'*wrf_
irf_TX32');
    tol12 = vrf_new12 - vrf_old12;

jt12 = jt12 + 1;
end

SE_IRF_TX32(SNRLoop,n_sistem)                   =
real(log2(det((eye(num_u_TX32))                 +
(1/variance_sistem)*Wt_irf_TX32*(inv(Wt_irf_TX32
'*Wt_irf_TX32))*Wt_irf_TX32'*H_irf_TX32*Vt_irf_T
X32*Vt_irf_TX32'*H_irf_TX32')));
```

```
%%%%%  MEMBUAT SISTEM DENGAN ANTENA TX=16

H_irf_TX16 =  H_irf(:,1:num_m_TX16);
F1_irf_TX16                                    =
H_irf_TX16(:,1:num_m_TX16)'*H_irf_TX16(:,1:num_m
_TX16);
gamma_irf_TX16                                 =
sqrt(P_sistem/(num_m_TX16*NRFt_HB));
vrf_irf_TX16       = ones(num_m_TX16,NRFt_HB);
l1_irf_TX16        = length(num_m_TX16);
nij_irf_TX16       = zeros(num_m_TX16,NRFt_HB);
vrf_irf1_TX16      = zeros(num_m_TX16,NRFt_HB);
vrf_new13         = 100;
tol13             = 1;
jt13              = 0;

while (tol13 > 1e-5)
    for j1_irf_TX16 = 1:1:NRFt_HB
        vrfj_IRF_TX16 = vrf_irf_TX16;
        vrfj_IRF_TX16(:,j1_irf_TX16)= [];
        I1_irf_TX16 = eye(NRFt_HB-1);
        C1_irf_TX16     =     I1_irf_TX16     +
((gamma_irf_TX16^2/variance_sistem)*vrfj_IRF_TX1
6'*F1_irf_TX16*vrfj_IRF_TX16);
        G1_irf_TX16                            =
((gamma_irf_TX16^2/variance_sistem)*F1_irf_TX16)
-
((gamma_irf_TX16^4/variance_sistem^2)*F1_irf_TX1
6*vrfj_IRF_TX16*(C1_irf_TX16^(-
1))*vrfj_IRF_TX16'*F1_irf_TX16);
        sigma_IRF_TX16 = 0;
        for i1_irf_TX16 = 1:1:num_m_TX16
            for L1_irf_TX16 = 1:1:num_m_TX16
                if L1_irf_TX16 ~= i1_irf_TX16
                sigma_IRF_TX16 = sigma_IRF_TX16 +
G1_irf_TX16(i1_irf_TX16,L1_irf_TX16)           *
vrf_irf_TX16(L1_irf_TX16,j1_irf_TX16);
                end
            end
            nij_irf_TX16=sigma_IRF_TX16;
```

```
            if nij_irf_TX16 == 0

vrf_irf_TX16(i1_irf_TX16,j1_irf_TX16) = 1;
            else

vrf_irf_TX16(i1_irf_TX16,j1_irf_TX16)           =
nij_irf_TX16/abs(nij_irf_TX16);
            end
        end
          cekv_FDB = abs(vrf_irf_TX16);
    end
    I2_irf_TX16 = eye(NRFr_HB);
    VdVdH_irf_16_TX16                           =
(sqrt(gamma_irf_TX16^2))*I2_irf_TX16;
    Heff_irf_TX16 = H_irf_TX16*vrf_irf_TX16;
    Q_irf_TX16 = vrf_irf_TX16'*vrf_irf_TX16;
    svd_irf_TX16  =  Heff_irf_TX16*(Q_irf_TX16^(-
0.5));
    [U_irf_TX16,S_irf_TX16,V_irf_TX16]=
svd(svd_irf_TX16);
    Ue_irf_TX16= V_irf_TX16(1:NRFt_HB,1:Ns_HB);
    Te_irf_TX16=
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_irf_TX16          =          (Q_irf_TX16^(-
0.5))*Ue_irf_TX16*Te_irf_TX16;

    vrf_old13 = vrf_new13;
    vrf_new13                                   =
trace(vrf_irf_TX16*Vd_irf_TX16*Vd_irf_TX16'*vrf_
irf_TX16');
    tol13 = vrf_new13 - vrf_old13;

    jt13 = jt13 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_TX16        = vrf_irf_TX16*Vd_irf_TX16;
F2_irf_TX16                                     =
```

```
H_irf_TX16*Vt_irf_TX16*Vt_irf_TX16'*H_irf_TX16';

wrf_irf_TX16        = ones(num_u_TX16,NRFr_HB);
l2_irf_TX16         = length(num_u_TX16);
nij2_irf_TX16       = zeros(num_u_TX16,NRFr_HB);
Wrf_irf1_TX16       = zeros(num_u_TX16,NRFr_HB);
vrf_new14 = 100;
tol14 = 1;
jt14 = 0;

while (tol14 > 1e-5)
    for j2_irf_TX16 = 1:1:NRFr_HB
        wrfj_IRF_TX16           = wrf_irf_TX16;
        wrfj_IRF_TX16(:,j2_irf_TX16)      = [];
        I2_irf_TX16                = eye(NRFr_HB-
1);
        C2_irf_TX16               = I2_irf_TX16 +
(((1/num_u_TX16)/variance_sistem)*wrfj_IRF_TX16'
*F2_irf_TX16*wrfj_IRF_TX16);
        G2_irf_TX16                                =
(((1/num_u_TX16)/variance_sistem)*F2_irf_TX16)  -
(((1/num_u_TX16)^2/variance_sistem^2)*F2_irf_TX1
6*wrfj_IRF_TX16*(C2_irf_TX16^(-
1))*wrfj_IRF_TX16'*F2_irf_TX16);
        sigma2_IRF_TX16 = 0;
        for i2_irf_TX16 = 1:1:num_u_TX16
            for L2_irf_TX16 = 1:1:num_u_TX16
                if L2_irf_TX16 ~= i2_irf_TX16
                sigma2_IRF_TX16 = sigma2_IRF_TX16
+     G2_irf_TX16(i2_irf_TX16,L2_irf_TX16)     *
wrf_irf_TX16(L2_irf_TX16,j2_irf_TX16);
                end
            end
            nij2_irf_TX16=sigma2_IRF_TX16;
            if nij2_irf_TX16 == 0

wrf_irf_TX16(i2_irf_TX16,j2_irf_TX16) = 1;
            else

wrf_irf_TX16(i2_irf_TX16,j2_irf_TX16)          =
```

```
nij2_irf_TX16/abs(nij2_irf_TX16);
            end
        end
    end
    j_IRF_TX16                                          =
wrf_irf_TX16'*H_irf_TX16*vrf_irf_TX16*Vd_irf_TX1
6*Vd_irf_TX16'*vrf_irf_TX16'*H_irf_TX16'*wrf_irf
_TX16                                                   +
variance_sistem*wrf_irf_TX16'*wrf_irf_TX16;
    J_IRF_TX16           = inv(j_IRF_TX16);
    Wd_irf_TX16                                         =
J_IRF_TX16*wrf_irf_TX16'*H_irf_TX16*vrf_irf_TX16
*Vd_irf_TX16;
    Wt_irf_TX16      = wrf_irf_TX16*Wd_irf_TX16;

    vrf_old14 = vrf_new14;
    vrf_new14                                           =
trace(wrf_irf_TX16*Wd_irf_TX16*Wd_irf_TX16'*wrf_
irf_TX16');
    tol14 = vrf_new14 - vrf_old14;

jt14 = jt14 + 1;
end

SE_IRF_TX16(SNRLoop,n_sistem)                           =
real(log2(det((eye(num_u_TX16))                         +
(1/variance_sistem)*Wt_irf_TX16*(inv(Wt_irf_TX16
'*Wt_irf_TX16))*Wt_irf_TX16'*H_irf_TX16*Vt_irf_T
X16*Vt_irf_TX16'*H_irf_TX16')));

%%%%% MEMBUAT SISTEM DENGAN ANTENA TX=64

F1_urlos_TX64        = H_urlos'*H_urlos;
gamma_urlos_TX64                                        =
sqrt(P_sistem/(num_m_TX64*NRFt_HB));
vrf_urlos_TX64       = ones(num_m_TX64,NRFt_HB);
l1_urlos_TX64        = length(num_m_TX64);
nij_urlos_TX64       = zeros(num_m_TX64,NRFt_HB);
vrf_urlos1_TX64      = zeros(num_m_TX64,NRFt_HB);
vrf_new15         = 100;
```

```
tol15              = 1;
jt15               = 0;

while (tol15 > 1e-5)
    for              j1_urlos_TX64              =
1:1:NRFt_HB                                % step 2
        vrfj_urlos_TX64 = vrf_urlos_TX64;
        vrfj_urlos_TX64(:,j1_urlos_TX64) = [];
        I1_urlos_TX64 = eye(NRFt_HB-1);
        C1_urlos_TX64    =    I1_urlos_TX64    +
((gamma_urlos_TX64^2/variance_sistem)*vrfj_urlos
_TX64'*F1_urlos_TX64*vrfj_urlos_TX64);
        G1_urlos_TX64                           =
((gamma_urlos_TX64^2/variance_sistem)*F1_urlos_T
X64)                                            -
((gamma_urlos_TX64^4/variance_sistem^2)*F1_urlos
_TX64*vrfj_urlos_TX64*(C1_urlos_TX64^(-
1))*vrfj_urlos_TX64'*F1_urlos_TX64);
        sigma_URLOS_TX64 = 0;
        for i1_urlos_TX64 = 1:1:num_m_TX64
            for L1_urlos_TX64 = 1:1:num_m_TX64
                if L1_urlos_TX64 ~= i1_urlos_TX64
                sigma_URLOS_TX64                =
sigma_URLOS_TX64                                +
G1_urlos_TX64(i1_urlos_TX64,L1_urlos_TX64)      *
vrf_urlos_TX64(L1_urlos_TX64,j1_urlos_TX64);
                end
            end
            nij_urlos_TX64=sigma_URLOS_TX64;
            if nij_urlos_TX64 == 0

vrf_urlos_TX64(i1_urlos_TX64,j1_urlos_TX64) = 1;
            else

vrf_urlos_TX64(i1_urlos_TX64,j1_urlos_TX64)     =
nij_urlos_TX64/abs(nij_urlos_TX64);
            end
        end
          cekv_FDB = abs(vrf_urlos_TX64);
    end
```

```
    I2_urlos_TX64= eye(NRFr_HB);
    VdVdh_urlos_TX64                                =
(sqrt(gamma_urlos_TX64^2))*I2_urlos_TX64;
    Heff_urlos_TX64 = H_urlos*vrf_urlos_TX64;
    Q_urlos_TX64=
vrf_urlos_TX64'*vrf_urlos_TX64;
    svd_urlos_TX64=
Heff_urlos_TX64*(Q_urlos_TX64^(-0.5));
    [U_urlos_TX64,S_urlos_TX64,V_urlos_TX64]
= svd(svd_urlos_TX64);
    Ue_urlos_TX64                                   =
V_urlos_TX64(1:NRFt_HB,1:Ns_HB);
    Te_urlos_TX64                                   =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_urlos_TX64       =       (Q_urlos_TX64^(-
0.5))*Ue_urlos_TX64*Te_urlos_TX64;

    vrf_old15 = vrf_new15;
    vrf_new15                                       =
trace(vrf_urlos_TX64*Vd_urlos_TX64*Vd_urlos_TX64
'*vrf_urlos_TX64');
    tol15 = vrf_new15 - vrf_old15;

    jt15 = jt15 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_TX64 = vrf_urlos_TX64*Vd_urlos_TX64;
F2_urlos_TX64                                       =
H_urlos*Vt_urlos_TX64*Vt_urlos_TX64'*H_urlos';

wrf_urlos_TX64= ones(num_u_TX64,NRFr_HB);
l2_urlos_TX64 = length(num_u_TX64);
nij2_urlos_TX64 = zeros(num_u_TX64,NRFr_HB);
Wrf_urlos1_TX64 = zeros(num_u_TX64,NRFr_HB);
vrf_new16 = 100;
tol16 = 1;
jt16 = 0;
```

```
while (tol16 > 1e-5)
    for j2_urlos_TX64 = 1:1:NRFr_HB
        wrfj_urlos_TX64         = wrf_urlos_TX64;
        wrfj_urlos_TX64(:,j2_urlos_TX64)        =
[];
        I2_urlos_TX64             = eye(NRFr_HB-
1);
        C2_urlos_TX64            = I2_urlos_TX64
+
(((1/num_u_TX64)/variance_sistem)*wrfj_urlos_TX6
4'*F2_urlos_TX64*wrfj_urlos_TX64);
        G2_urlos_TX64                           =
(((1/num_u_TX64)/variance_sistem)*F2_urlos_TX64)
-
(((1/num_u_TX64)^2/variance_sistem^2)*F2_urlos_T
X64*wrfj_urlos_TX64*(C2_urlos_TX64^(-
1))*wrfj_urlos_TX64'*F2_urlos_TX64);
        sigma2_urlos_TX64 = 0;
        for i2_urlos_TX64 = 1:1:num_u_TX64
            for L2_urlos_TX64 = 1:1:num_u_TX64
                if L2_urlos_TX64 ~= i2_urlos_TX64
                sigma2_urlos_TX64               =
sigma2_urlos_TX64                               +
G2_urlos_TX64(i2_urlos_TX64,L2_urlos_TX64)      *
wrf_urlos_TX64(L2_urlos_TX64,j2_urlos_TX64);
                end
            end
            nij2_urlos_TX64=sigma2_urlos_TX64;
            if nij2_urlos_TX64 == 0

wrf_urlos_TX64(i2_urlos_TX64,j2_urlos_TX64) = 1;
            else

wrf_urlos_TX64(i2_urlos_TX64,j2_urlos_TX64)     =
nij2_urlos_TX64/abs(nij2_urlos_TX64);
            end
        end
    end
    j_urlos_TX64                                =
```

```
wrf_urlos_TX64'*H_urlos*vrf_urlos_TX64*Vd_urlos_
TX64*Vd_urlos_TX64'*vrf_urlos_TX64'*H_urlos'*wrf
_urlos_TX64                                      +
variance_sistem*wrf_urlos_TX64'*wrf_urlos_TX64;
    J_urlos_TX64          = inv(j_urlos_TX64);
    Wd_urlos_TX64                                =
J_urlos_TX64*wrf_urlos_TX64'*H_urlos*vrf_urlos_T
X64*Vd_urlos_TX64;
    Wt_urlos_TX64                                =
wrf_urlos_TX64*Wd_urlos_TX64;

    vrf_old16 = vrf_new16;
    vrf_new16                                    =
trace(wrf_urlos_TX64*Wd_urlos_TX64*Wd_urlos_TX64
'*wrf_urlos_TX64');
    tol16 = vrf_old16 - vrf_new16;

jt16 = jt16 + 1;
end

SE_urlos_TX64(SNRLoop,n_sistem)                  =
real(log2(det((eye(num_u_TX64))                  +
(1/variance_sistem)*Wt_urlos_TX64*(inv(Wt_urlos_
TX64'*Wt_urlos_TX64))*Wt_urlos_TX64'*H_urlos*Vt_
urlos_TX64*Vt_urlos_TX64'*H_urlos')));

%%%%% MEMBUAT SISTEM DENGAN ANTENA TX=32

H_urlos_TX32 =  H_urlos(:,1:num_m_TX32);
F1_urlos_TX32                                    =
H_urlos_TX32(:,1:num_m_TX32)'*H_urlos_TX32(:,1:n
um_m_TX32);
gamma_urlos_TX32                                 =
sqrt(P_sistem/(num_m_TX32*NRFt_HB));
vrf_urlos_TX32       = ones(num_m_TX32,NRFt_HB);
l1_urlos_TX32        = length(num_m_TX32);
nij_urlos_TX32       = zeros(num_m_TX32,NRFt_HB);
vrf_urlos1_TX32      = zeros(num_m_TX32,NRFt_HB);
vrf_new17         = 100;
tol17             = 1;
```

```
jt17                = 0;

while (tol17 > 1e-5)
    for j1_urlos_TX32=1:1:NRFt_HB
        vrfj_URLOS_TX32=vrf_urlos_TX32;
        vrfj_URLOS_TX32(:,j1_urlos_TX32)  = [];
        I1_urlos_TX32=eye(NRFt_HB-1);
        C1_urlos_TX32=I1_urlos_TX32             +
((gamma_urlos_TX32^2/variance_sistem)*vrfj_URLOS
_TX32'*F1_urlos_TX32*vrfj_URLOS_TX32);          %
step 3
        G1_urlos_TX32                           =
((gamma_urlos_TX32^2/variance_sistem)*F1_urlos_T
X32)                                            -
((gamma_urlos_TX32^4/variance_sistem^2)*F1_urlos
_TX32*vrfj_URLOS_TX32*(C1_urlos_TX32^(-
1))*vrfj_URLOS_TX32'*F1_urlos_TX32);    % step 4
        sigma_URLOS_TX32 = 0;
        for            i1_urlos_TX32            =
1:1:num_m_TX32                          % step 5
            %sigma_TX32 = 0;
            for L1_urlos_TX32 = 1:1:num_m_TX32
                %sigma_TX32 = 0;
                if L1_urlos_TX32 ~= i1_urlos_TX32
                sigma_URLOS_TX32                =
sigma_URLOS_TX32                                +
G1_urlos_TX32(i1_urlos_TX32,L1_urlos_TX32)      *
vrf_urlos_TX32(L1_urlos_TX32,j1_urlos_TX32);
    % step 6
                end
            end
            nij_urlos_TX32=sigma_URLOS_TX32;
            if          nij_urlos_TX32          ==
0                         % step 7

vrf_urlos_TX32(i1_urlos_TX32,j1_urlos_TX32) = 1;
            else

vrf_urlos_TX32(i1_urlos_TX32,j1_urlos_TX32)     =
nij_urlos_TX32/abs(nij_urlos_TX32);
```

```
            end
        end
          cekv_FDB = abs(vrf_urlos_TX32);
    end
    I2_urlos_TX32                 = eye(NRFr_HB);
    VdVdH_urlos_32_TX32                          =
(sqrt(gamma_urlos_TX32^2))*I2_urlos_TX32;
    Heff_urlos_TX32                              =
H_urlos_TX32*vrf_urlos_TX32;
    Q_urlos_TX32                                 =
vrf_urlos_TX32'*vrf_urlos_TX32;
    svd_urlos_TX32                               =
Heff_urlos_TX32*(Q_urlos_TX32^(-0.5));
    [U_urlos_TX32,S_urlos_TX32,V_urlos_TX32]
= svd(svd_urlos_TX32);
    Ue_urlos_TX32                                =
V_urlos_TX32(1:NRFt_HB,1:Ns_HB);
    Te_urlos_TX32                                =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_urlos_TX32              = (Q_urlos_TX32^(-
0.5))*Ue_urlos_TX32*Te_urlos_TX32;

    vrf_old17 = vrf_new17;
    vrf_new17                                    =
trace(vrf_urlos_TX32*Vd_urlos_TX32*Vd_urlos_TX32
'*vrf_urlos_TX32');
    tol17 = vrf_new17 - vrf_old17;

    jt17 = jt17 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_TX32                                    =
vrf_urlos_TX32*Vd_urlos_TX32;
F2_urlos_TX32                                    =
H_urlos_TX32*Vt_urlos_TX32*Vt_urlos_TX32'*H_urlo
s_TX32';
```

```
%P = SNR;
wrf_urlos_TX32                                          =
ones(num_u_TX32,NRFr_HB);           %step 1
l2_urlos_TX32          = length(num_u_TX32);
nij2_urlos_TX32        = zeros(num_u_TX32,NRFr_HB);
Wrf_urlos1_TX32        = zeros(num_u_TX32,NRFr_HB);
vrf_new18 = 100;
tol18 = 1;
jt18 = 0;

while (tol18 > 1e-5)
    for              j2_urlos_TX32                  =
1:1:NRFr_HB                                        %step
2
        wrfj_URLOS_TX32          = wrf_urlos_TX32;
        wrfj_URLOS_TX32(:,j2_urlos_TX32)           =
[];
        I2_urlos_TX32               = eye(NRFr_HB-
1);   %step 3&4
        C2_urlos_TX32              = I2_urlos_TX32
+
(((1/num_u_TX32)/variance_sistem)*wrfj_URLOS_TX3
2'*F2_urlos_TX32*wrfj_URLOS_TX32);
        G2_urlos_TX32                              =
(((1/num_u_TX32)/variance_sistem)*F2_urlos_TX32)
-
(((1/num_u_TX32)^2/variance_sistem^2)*F2_urlos_T
X32*wrfj_URLOS_TX32*(C2_urlos_TX32^(-
1))*wrfj_URLOS_TX32'*F2_urlos_TX32);
        sigma2_URLOS_TX32 = 0;
        for i2_urlos_TX32 = 1:1:num_u_TX32
            %sigma2_TX32 = 0;
            for L2_urlos_TX32 = 1:1:num_u_TX32
                %sigma2_TX32 = 0;
                if L2_urlos_TX32 ~= i2_urlos_TX32
                sigma2_URLOS_TX32                  =
sigma2_URLOS_TX32                                  +
G2_urlos_TX32(i2_urlos_TX32,L2_urlos_TX32)         *
wrf_urlos_TX32(L2_urlos_TX32,j2_urlos_TX32);
                end
```

```
            end
            nij2_urlos_TX32=sigma2_URLOS_TX32;
            if nij2_urlos_TX32 == 0

wrf_urlos_TX32(i2_urlos_TX32,j2_urlos_TX32) = 1;
            else

wrf_urlos_TX32(i2_urlos_TX32,j2_urlos_TX32)     =
nij2_urlos_TX32/abs(nij2_urlos_TX32);
            end
            %i2_urlos_TX32 = i2_urlos + 1;
        end
      %j2_urlos_TX32 = j2_urlos + 1;
    end
    j_urlos_TX32                                    =
wrf_urlos_TX32'*H_urlos_TX32*vrf_urlos_TX32*Vd_u
rlos_TX32*Vd_urlos_TX32'*vrf_urlos_TX32'*H_urlos
_TX32'*wrf_urlos_TX32                               +
variance_sistem*wrf_urlos_TX32'*wrf_urlos_TX32;
    J_urlos_TX32           = inv(j_urlos_TX32);
    Wd_urlos_TX32                                   =
J_urlos_TX32*wrf_urlos_TX32'*H_urlos_TX32*vrf_ur
los_TX32*Vd_urlos_TX32;
    Wt_urlos_TX32                                   =
wrf_urlos_TX32*Wd_urlos_TX32;

    vrf_old18 = vrf_new18;
    vrf_new18                                       =
trace(wrf_urlos_TX32*Wd_urlos_TX32*Wd_urlos_TX32
'*wrf_urlos_TX32');
    tol18 = vrf_old18 - vrf_new18;

jt18 = jt18 + 1;
end

SE_urlos_TX32(SNRLoop,n_sistem)                     =
real(log2(det((eye(num_u_TX32))                     +
(1/variance_sistem)*Wt_urlos_TX32*(inv(Wt_urlos_
TX32'*Wt_urlos_TX32))*Wt_urlos_TX32'*H_urlos_TX3
2*Vt_urlos_TX32*Vt_urlos_TX32'*H_urlos_TX32')));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% MEMBUAT SISTEM DENGAN ANTENA TX=16

H_urlos_16 =  H_urlos(:,1:num_m_TX16);
F1_urlos_TX16                                           =
H_urlos_16(:,1:num_m_TX16)'*H_urlos_16(:,1:num_m
_TX16);
gamma_urlos_TX16                                        =
sqrt(P_sistem/(num_m_TX16*NRFt_HB));
vrf_urlos_TX16                                          =
ones(num_m_TX16,NRFt_HB);                      % step 1
l1_urlos_TX16          = length(num_m_TX16);
nij_urlos_TX16         = zeros(num_m_TX16,NRFt_HB);
vrf_urlos1_TX16        = zeros(num_m_TX16,NRFt_HB);
vrf_new19           = 100;
tol19               = 1;
jt19                = 0;

while (tol19 > 1e-5)
    for             j1_urlos_TX16                       =
1:1:NRFt_HB                                  % step 2
        vrfj_URLOS_TX16                                 =
vrf_urlos_TX16;
        vrfj_URLOS_TX16(:,j1_urlos_TX16)                =
[];
        I1_urlos_TX16              = eye(NRFt_HB-
1);
        C1_urlos_TX16             = I1_urlos_TX16
+
((gamma_urlos_TX16^2/variance_sistem)*vrfj_URLOS
_TX16'*F1_urlos_TX16*vrfj_URLOS_TX16);          %
step 3
        G1_urlos_TX16                                   =
((gamma_urlos_TX16^2/variance_sistem)*F1_urlos_T
X16)                                                    -
((gamma_urlos_TX16^4/variance_sistem^2)*F1_urlos
_TX16*vrfj_URLOS_TX16*(C1_urlos_TX16^(-
1))*vrfj_URLOS_TX16'*F1_urlos_TX16);    % step 4
```

```
        sigma_URLOS_TX16 = 0;
        for            i1_urlos_TX16            =
1:1:num_m_TX16                          % step 5
            %sigma_TX16 = 0;
            for L1_urlos_TX16 = 1:1:num_m_TX16
                %sigma_TX16 = 0;
                if L1_urlos_TX16 ~= i1_urlos_TX16
                sigma_URLOS_TX16                 =
sigma_URLOS_TX16                                 +
G1_urlos_TX16(i1_urlos_TX16,L1_urlos_TX16)       *
vrf_urlos_TX16(L1_urlos_TX16,j1_urlos_TX16);
    % step 6
                end
            end
            nij_urlos_TX16=sigma_URLOS_TX16;
            if          nij_urlos_TX16          ==
0                         % step 7

vrf_urlos_TX16(i1_urlos_TX16,j1_urlos_TX16) = 1;
            else

vrf_urlos_TX16(i1_urlos_TX16,j1_urlos_TX16)      =
nij_urlos_TX16/abs(nij_urlos_TX16);
            end
        end
          cekv_FDB = abs(vrf_urlos_TX16);
    end
    I2_urlos_TX16                  = eye(NRFr_HB);
    VdVdH_urlos_16_TX16                          =
(sqrt(gamma_urlos_TX16^2))*I2_urlos_TX16;
    Heff_urlos_TX16                              =
H_urlos_16*vrf_urlos_TX16;
    Q_urlos_TX16                                 =
vrf_urlos_TX16'*vrf_urlos_TX16;
    svd_urlos_TX16                               =
Heff_urlos_TX16*(Q_urlos_TX16^(-0.5));
    [U_urlos_TX16,S_urlos_TX16,V_urlos_TX16]
= svd(svd_urlos_TX16);
    Ue_urlos_TX16                                =
V_urlos_TX16(1:NRFt_HB,1:Ns_HB);
```

```
    Te_urlos_TX16                                    =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_urlos_TX16              = (Q_urlos_TX16^(-
0.5))*Ue_urlos_TX16*Te_urlos_TX16;

    vrf_old19 = vrf_new19;
    vrf_new19                                        =
trace(vrf_urlos_TX16*Vd_urlos_TX16*Vd_urlos_TX16
'*vrf_urlos_TX16');
    tol19 = vrf_new19 - vrf_old19;

    jt19 = jt19 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_TX16                                        =
vrf_urlos_TX16*Vd_urlos_TX16;
F2_urlos_TX16                                        =
H_urlos_16*Vt_urlos_TX16*Vt_urlos_TX16'*H_urlos_
16';

wrf_urlos_TX16                                       =
ones(num_u_TX16,NRFr_HB);          %step 1
l2_urlos_TX16         = length(num_u_TX16);
nij2_urlos_TX16       = zeros(num_u_TX16,NRFr_HB);
Wrf_urlos1_TX16       = zeros(num_u_TX16,NRFr_HB);
vrf_new20 = 100;
tol20 = 1;
jt20 = 0;

while (tol20 > 1e-5)
    for              j2_urlos_TX16               =
1:1:NRFr_HB                                    %step
2
        wrfj_URLOS_TX16          = wrf_urlos_TX16;
        wrfj_URLOS_TX16(:,j2_urlos_TX16)         =
[];
        I2_urlos_TX16               = eye(NRFr_HB-
```

```
1);   %step 3&4
        C2_urlos_TX16              = I2_urlos_TX16
+
(((1/num_u_TX16)/variance_sistem)*wrfj_URLOS_TX1
6'*F2_urlos_TX16*wrfj_URLOS_TX16);
        G2_urlos_TX16                              =
(((1/num_u_TX16)/variance_sistem)*F2_urlos_TX16)
-
(((1/num_u_TX16)^2/variance_sistem^2)*F2_urlos_T
X16*wrfj_URLOS_TX16*(C2_urlos_TX16^(-
1))*wrfj_URLOS_TX16'*F2_urlos_TX16);
        sigma2_URLOS_TX16 = 0;
        for i2_urlos_TX16 = 1:1:num_u_TX16
            %sigma2_TX16 = 0;
            for L2_urlos_TX16 = 1:1:num_u_TX16
                %sigma2_TX16 = 0;
                if L2_urlos_TX16 ~= i2_urlos_TX16
                sigma2_URLOS_TX16                =
sigma2_URLOS_TX16                                +
G2_urlos_TX16(i2_urlos_TX16,L2_urlos_TX16)       *
wrf_urlos_TX16(L2_urlos_TX16,j2_urlos_TX16);
                end
            end
            nij2_urlos_TX16=sigma2_URLOS_TX16;
            if nij2_urlos_TX16 == 0

wrf_urlos_TX16(i2_urlos_TX16,j2_urlos_TX16) = 1;
            else

wrf_urlos_TX16(i2_urlos_TX16,j2_urlos_TX16)      =
nij2_urlos_TX16/abs(nij2_urlos_TX16);
            end
            %i2_urlos_TX16 = i2_urlos + 1;
        end
      %j2_urlos_TX16 = j2_urlos + 1;
    end
    j_urlos_TX16                                   =
wrf_urlos_TX16'*H_urlos_16*vrf_urlos_TX16*Vd_url
os_TX16*Vd_urlos_TX16'*vrf_urlos_TX16'*H_urlos_1
6'*wrf_urlos_TX16                                  +
```

```
variance_sistem*wrf_urlos_TX16'*wrf_urlos_TX16;
    J_urlos_TX16           = inv(j_urlos_TX16);
    Wd_urlos_TX16                                =
J_urlos_TX16*wrf_urlos_TX16'*H_urlos_16*vrf_urlo
s_TX16*Vd_urlos_TX16;
    Wt_urlos_TX16                                =
wrf_urlos_TX16*Wd_urlos_TX16;

    vrf_old20 = vrf_new20;
    vrf_new20                                    =
trace(wrf_urlos_TX16*Wd_urlos_TX16*Wd_urlos_TX16
'*wrf_urlos_TX16');
    tol20 = vrf_new20 - vrf_old20;

jt20 = jt20 + 1;
end

SE_urlos_TX16(SNRLoop,n_sistem)                  =
real(log2(det((eye(num_u_TX16))                  +
(1/variance_sistem)*Wt_urlos_TX16*(inv(Wt_urlos_
TX16'*Wt_urlos_TX16))*Wt_urlos_TX16'*H_urlos_16*
Vt_urlos_TX16*Vt_urlos_TX16'*H_urlos_16')));

%%%%% MEMBUAT SISTEM DENGAN ANTENA RX=64 pada
kanal IRF
F1_irf_RX64        = H_irf'*H_irf;
gamma_irf_RX64                                   =
sqrt(P_sistem/(num_m_RX64*NRFt_HB));
vrf_irf_RX64                                     =
ones(num_m_RX64,NRFt_HB);                % step 1
l1_irf_RX64        = length(num_m_RX64);
nij_irf_RX64       = zeros(num_m_RX64,NRFt_HB);
vrf_irf1_RX64      = zeros(num_m_RX64,NRFt_HB);
vrf_new21           = 100;
tol21               = 1;
jt21                = 0;

while (tol21 > 1e-5)
    for              j1_irf_RX64                 =
1:1:NRFt_HB                              % step 2
```

```
        vrfj_irf_RX64              = vrf_irf_RX64;
        vrfj_irf_RX64(:,j1_irf_RX64)       = [];
        I1_irf_RX64                 = eye(NRFt_HB-
1);
        C1_irf_RX64                = I1_irf_RX64 +
((gamma_irf_RX64^2/variance_sistem)*vrfj_irf_RX6
4'*F1_irf_RX64*vrfj_irf_RX64);           % step 3
        G1_irf_RX64                                =
((gamma_irf_RX64^2/variance_sistem)*F1_irf_RX64)
-
((gamma_irf_RX64^4/variance_sistem^2)*F1_irf_RX6
4*vrfj_irf_RX64*(C1_irf_RX64^(-
1))*vrfj_irf_RX64'*F1_irf_RX64);   % step 4
        sigma_irf_RX64 = 0;
        for              i1_irf_RX64              =
1:1:num_m_RX64                        % step 5
            for L1_irf_RX64 = 1:1:num_m_RX64
                if L1_irf_RX64 ~= i1_irf_RX64
                sigma_irf_RX64 = sigma_irf_RX64 +
G1_irf_RX64(i1_irf_RX64,L1_irf_RX64)             *
vrf_irf_RX64(L1_irf_RX64,j1_irf_RX64);
% step 6
                end
            end
            nij_irf_RX64 = sigma_irf_RX64;
            if           nij_irf_RX64          ==
0                         % step 7

vrf_irf_RX64(i1_irf_RX64,j1_irf_RX64) = 1;
            else

vrf_irf_RX64(i1_irf_RX64,j1_irf_RX64)           =
nij_irf_RX64/abs(nij_irf_RX64);
            end
        end
          cekv_FDB = abs(vrf_irf_RX64);
    end
    I2_irf_RX64                    = eye(NRFr_HB);
    VdVdh_irf_RX64                                 =
(sqrt(gamma_irf_RX64^2))*I2_irf_RX64;
```

```
    Heff_irf_RX64                                   =
H_irf*vrf_irf_RX64;
    Q_irf_RX64                                      =
vrf_irf_RX64'*vrf_irf_RX64;
    svd_irf_RX64                                    =
Heff_irf_RX64*(Q_irf_RX64^(-0.5));
    [U_irf_RX64,S_irf_RX64,V_irf_RX64]              =
svd(svd_irf_RX64);
    Ue_irf_RX64                                     =
V_irf_RX64(1:NRFt_HB,1:Ns_HB);
    Te_irf_RX64                                     =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_irf_RX64                  = (Q_irf_RX64^(-
0.5))*Ue_irf_RX64*Te_irf_RX64;

    vrf_old21 = vrf_new21;
    vrf_new21                                       =
trace(vrf_irf_RX64*Vd_irf_RX64*Vd_irf_RX64'*vrf_
irf_RX64');
    tol21 = vrf_new21 - vrf_old21;

    jt21 = jt21 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_RX64                                         =
vrf_irf_RX64*Vd_irf_RX64;
F2_irf_RX64                                         =
H_irf*Vt_irf_RX64*Vt_irf_RX64'*H_irf';

wrf_irf_RX64                                        =
ones(num_u_RX64,NRFr_HB);          %step 1
l2_irf_RX64         = length(num_u_RX64);
nij2_irf_RX64       = zeros(num_u_RX64,NRFr_HB);
Wrf_irf1_RX64       = zeros(num_u_RX64,NRFr_HB);
vrf_new22 = 100;
tol22 = 1;
jt22 = 0;
```

```
while (tol22 > 1e-5)
    for                j2_irf_RX64                =
1:1:NRFr_HB                                      %step
2
        wrfj_irf_RX64             = wrf_irf_RX64;
        wrfj_irf_RX64(:,j2_irf_RX64)       = [];
        I2_irf_RX64                 = eye(NRFr_HB-
1);   %step 3&4
        C2_irf_RX64               = I2_irf_RX64 +
(((1/num_u_RX64)/variance_sistem)*wrfj_irf_RX64'
*F2_irf_RX64*wrfj_irf_RX64);
        G2_irf_RX64                                =
(((1/num_u_RX64)/variance_sistem)*F2_irf_RX64)  -
(((1/num_u_RX64)^2/variance_sistem^2)*F2_irf_RX6
4*wrfj_irf_RX64*(C2_irf_RX64^(-
1))*wrfj_irf_RX64'*F2_irf_RX64);
        sigma2_irf_RX64 = 0;
        for i2_irf_RX64 = 1:1:num_u_RX64
            for L2_irf_RX64 = 1:1:num_u_RX64
                if L2_irf_RX64 ~= i2_irf_RX64
                sigma2_irf_RX64 = sigma2_irf_RX64
+      G2_irf_RX64(i2_irf_RX64,L2_irf_RX64)      *
wrf_irf_RX64(L2_irf_RX64,j2_irf_RX64);
                end
            end
            nij2_irf_RX64=sigma2_irf_RX64;
            if nij2_irf_RX64 == 0

wrf_irf_RX64(i2_irf_RX64,j2_irf_RX64) = 1;
            else

wrf_irf_RX64(i2_irf_RX64,j2_irf_RX64)           =
nij2_irf_RX64/abs(nij2_irf_RX64);
            end
            %i2_irf_RX64 = i2_irf + 1;
        end
      %j2_irf_RX64 = j2_irf + 1;
    end
    j_IRF_RX64                                  =
```

```
wrf_irf_RX64'*H_irf*vrf_irf_RX64*Vd_irf_RX64*Vd_
irf_RX64'*vrf_irf_RX64'*H_irf'*wrf_irf_RX64     +
variance_sistem*wrf_irf_RX64'*wrf_irf_RX64;
    J_IRF_RX64           = inv(j_IRF_RX64);
    Wd_irf_RX64                                 =
J_IRF_RX64*wrf_irf_RX64'*H_irf*vrf_irf_RX64*Vd_i
rf_RX64;
    Wt_irf_RX64      = wrf_irf_RX64*Wd_irf_RX64;

    vrf_old22 = vrf_new22;
    vrf_new22                                   =
trace(wrf_irf_RX64*Wd_irf_RX64*Wd_irf_RX64'*wrf_
irf_RX64');
    tol22 = vrf_new22 - vrf_old22;

jt22 = jt22 + 1;
end

SE_IRF_RX64(SNRLoop,n_sistem)                   =
real(log2(det((eye(num_u_RX64))                 +
(1/variance_sistem)*Wt_irf_RX64*(inv(Wt_irf_RX64
'*Wt_irf_RX64))*Wt_irf_RX64'*H_irf*Vt_irf_RX64*V
t_irf_RX64'*H_irf')));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% MEMBUAT SISTEM DENGAN ANTENA RX=32 pada
kanal IRF

H_irf_RX32           =  H_irf(1:num_u_RX32,:);
F1_irf_RX32                                     =
H_irf_RX32(:,1:num_m_RX32)'*H_irf_RX32(:,1:num_m
_RX32);
gamma_irf_RX32                                  =
sqrt(P_sistem/(num_m_RX32*NRFt_HB));
vrf_irf_RX32                                    =
ones(num_m_RX32,NRFt_HB);                % step 1
l1_irf_RX32        = length(num_m_RX32);
nij_irf_RX32       = zeros(num_m_RX32,NRFt_HB);
vrf_irf1_RX32      = zeros(num_m_RX32,NRFt_HB);
```

```
vrf_new23           = 100;
tol23               = 1;
jt23                = 0;

while (tol23 > 1e-5)
    for               j1_irf_RX32               =
1:1:NRFt_HB                                 % step 2
        vrfj_irf_RX32             = vrf_irf_RX32;
        vrfj_irf_RX32(:,j1_irf_RX32)      = [];
        I1_irf_RX32               = eye(NRFt_HB-
1);
        C1_irf_RX32               = I1_irf_RX32 +
((gamma_irf_RX32^2/variance_sistem)*vrfj_irf_RX3
2'*F1_irf_RX32*vrfj_irf_RX32);          % step 3
        G1_irf_RX32                             =
((gamma_irf_RX32^2/variance_sistem)*F1_irf_RX32)
-
((gamma_irf_RX32^4/variance_sistem^2)*F1_irf_RX3
2*vrfj_irf_RX32*(C1_irf_RX32^(-
1))*vrfj_irf_RX32'*F1_irf_RX32);   % step 4
        sigma_irf_RX32 = 0;
        for             i1_irf_RX32             =
1:1:num_m_RX32                          % step 5
            for L1_irf_RX32 = 1:1:num_m_RX32
                if L1_irf_RX32 ~= i1_irf_RX32
                sigma_irf_RX32 = sigma_irf_RX32 +
G1_irf_RX32(i1_irf_RX32,L1_irf_RX32)            *
vrf_irf_RX32(L1_irf_RX32,j1_irf_RX32);
% step 6
                end
            end
            nij_irf_RX32=sigma_irf_RX32;
            if          nij_irf_RX32          ==
0                        % step 7

vrf_irf_RX32(i1_irf_RX32,j1_irf_RX32) = 1;
            else

vrf_irf_RX32(i1_irf_RX32,j1_irf_RX32)          =
nij_irf_RX32/abs(nij_irf_RX32);
```

```
            end
        end
          cekv_FDB = abs(vrf_irf_RX32);
    end
    I2_irf_RX32                    = eye(NRFr_HB);
    VdVdH_irf_32_RX32                                =
(sqrt(gamma_irf_RX32^2))*I2_irf_RX32;
    Heff_irf_RX32                                    =
H_irf_RX32*vrf_irf_RX32;
    Q_irf_RX32                                       =
vrf_irf_RX32'*vrf_irf_RX32;
    svd_irf_RX32                                     =
Heff_irf_RX32*(Q_irf_RX32^(-0.5));
    [U_irf_RX32,S_irf_RX32,V_irf_RX32]               =
svd(svd_irf_RX32);
    Ue_irf_RX32                                      =
V_irf_RX32(1:NRFt_HB,1:Ns_HB);
    Te_irf_RX32                                      =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_irf_RX32                    = (Q_irf_RX32^(-
0.5))*Ue_irf_RX32*Te_irf_RX32;

    vrf_old23 = vrf_new23;
    vrf_new23                                        =
trace(vrf_irf_RX32*Vd_irf_RX32*Vd_irf_RX32'*vrf_
irf_RX32');
    tol23 = vrf_new23 - vrf_old23;

    jt23 = jt23 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_RX32                                          =
vrf_irf_RX32*Vd_irf_RX32;
F2_irf_RX32                                          =
H_irf_RX32*Vt_irf_RX32*Vt_irf_RX32'*H_irf_RX32';

wrf_irf_RX32                                         =
```

```
ones(num_u_RX32,NRFr_HB);          %step 1
l2_irf_RX32         = length(num_u_RX32);
nij2_irf_RX32       = zeros(num_u_RX32,NRFr_HB);
Wrf_irf1_RX32       = zeros(num_u_RX32,NRFr_HB);
vrf_new24 = 100;
tol24 = 1;
jt24 = 0;

while (tol24 > 1e-5)
    for               j2_irf_RX32               =
1:1:NRFr_HB                                    %step
2
        wrfj_IRF_RX32           = wrf_irf_RX32;
        wrfj_IRF_RX32(:,j2_irf_RX32)       = [];
        I2_irf_RX32                = eye(NRFr_HB-
1);   %step 3&4
        C2_irf_RX32               = I2_irf_RX32 +
(((1/num_u_RX32)/variance_sistem)*wrfj_IRF_RX32'
*F2_irf_RX32*wrfj_IRF_RX32);
        G2_irf_RX32                                =
(((1/num_u_RX32)/variance_sistem)*F2_irf_RX32)  -
(((1/num_u_RX32)^2/variance_sistem^2)*F2_irf_RX3
2*wrfj_IRF_RX32*(C2_irf_RX32^(-
1))*wrfj_IRF_RX32'*F2_irf_RX32);
        sigma2_RX32 = 0;
        for i2_irf_RX32 = 1:1:num_u_RX32
            for L2_irf_RX32 = 1:1:num_u_RX32
                if L2_irf_RX32 ~= i2_irf_RX32
                sigma2_RX32   =   sigma2_RX32   +
G2_irf_RX32(i2_irf_RX32,L2_irf_RX32)            *
wrf_irf_RX32(L2_irf_RX32,j2_irf_RX32);
                end
            end
            nij2_irf_RX32=sigma2_RX32;
            if nij2_irf_RX32 == 0

wrf_irf_RX32(i2_irf_RX32,j2_irf_RX32) = 1;
            else

wrf_irf_RX32(i2_irf_RX32,j2_irf_RX32)          =
```

```
nij2_irf_RX32/abs(nij2_irf_RX32);
            end
            %i2_irf_RX32 = i2_irf + 1;
        end
        %j2_irf_RX32 = j2_irf + 1;
    end
    j_IRF_RX32                                       =
wrf_irf_RX32'*H_irf_RX32*vrf_irf_RX32*Vd_irf_RX3
2*Vd_irf_RX32'*vrf_irf_RX32'*H_irf_RX32'*wrf_irf
_RX32                                            +
variance_sistem*wrf_irf_RX32'*wrf_irf_RX32;
    J_IRF_RX32           = inv(j_IRF_RX32);
    Wd_irf_RX32                                      =
J_IRF_RX32*wrf_irf_RX32'*H_irf_RX32*vrf_irf_RX32
*Vd_irf_RX32;
    Wt_irf_RX32       = wrf_irf_RX32*Wd_irf_RX32;

    vrf_old24 = vrf_new24;
    vrf_new24                                        =
trace(wrf_irf_RX32*Wd_irf_RX32*Wd_irf_RX32'*wrf_
irf_RX32');
    tol24 = vrf_new24 - vrf_old24;

jt24 = jt24 + 1;
end

SE_IRF_RX32(SNRLoop,n_sistem)                        =
real(log2(det((eye(num_u_RX32))                      +
(1/variance_sistem)*Wt_irf_RX32*(inv(Wt_irf_RX32
'*Wt_irf_RX32))*Wt_irf_RX32'*H_irf_RX32*Vt_irf_R
X32*Vt_irf_RX32'*H_irf_RX32')));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% SISTEM DENGAN RX = 16 pada kanal IRF

H_irf_RX16           =  H_irf(1:num_u_RX16,:);
F1_irf_RX16                                          =
H_irf_RX16(:,1:num_m_RX16)'*H_irf_RX16(:,1:num_m
_RX16);
```

```
gamma_irf_RX16                                    =
sqrt(P_sistem/(num_m_RX16*NRFt_HB));
vrf_irf_RX16                                      =
ones(num_m_RX16,NRFt_HB);                % step 1
l1_irf_RX16          = length(num_m_RX16);
nij_irf_RX16         = zeros(num_m_RX16,NRFt_HB);
vrf_irf1_RX16        = zeros(num_m_RX16,NRFt_HB);
vrf_new25            = 100;
tol25                = 1;
jt25                 = 0;

while (tol25 > 1e-5)
    for                j1_irf_RX16                =
1:1:NRFt_HB                                % step 2
        vrfj_irf_RX16              = vrf_irf_RX16;
        vrfj_irf_RX16(:,j1_irf_RX16)       = [];
        I1_irf_RX16                 = eye(NRFt_HB-
1);
        C1_irf_RX16                = I1_irf_RX16 +
((gamma_irf_RX16^2/variance_sistem)*vrfj_irf_RX1
6'*F1_irf_RX16*vrfj_irf_RX16);             % step 3
        G1_irf_RX16                                =
((gamma_irf_RX16^2/variance_sistem)*F1_irf_RX16)
-
((gamma_irf_RX16^4/variance_sistem^2)*F1_irf_RX1
6*vrfj_irf_RX16*(C1_irf_RX16^(-
1))*vrfj_irf_RX16'*F1_irf_RX16);   % step 4
        sigma_irf_RX16 = 0;
        for              i1_irf_RX16               =
1:1:num_m_RX16                             % step 5
            for L1_irf_RX16 = 1:1:num_m_RX16
                if L1_irf_RX16 ~= i1_irf_RX16
                sigma_irf_RX16 = sigma_irf_RX16 +
G1_irf_RX16(i1_irf_RX16,L1_irf_RX16)             *
vrf_irf_RX16(L1_irf_RX16,j1_irf_RX16);
% step 6
                end
            end
            nij_irf_RX16=sigma_irf_RX16;
            if           nij_irf_RX16          ==
```

```
0                          % step 7

vrf_irf_RX16(i1_irf_RX16,j1_irf_RX16) = 1;
            else

vrf_irf_RX16(i1_irf_RX16,j1_irf_RX16)           =
nij_irf_RX16/abs(nij_irf_RX16);
            end
        end
          cekv_FDB = abs(vrf_irf_RX16);
    end
    I2_irf_RX16                    = eye(NRFr_HB);
    VdVdH_irf_16_RX16                                =
(sqrt(gamma_irf_RX16^2))*I2_irf_RX16;
    Heff_irf_RX16                                    =
H_irf_RX16*vrf_irf_RX16;
    Q_irf_RX16                                       =
vrf_irf_RX16'*vrf_irf_RX16;
    svd_irf_RX16                                     =
Heff_irf_RX16*(Q_irf_RX16^(-0.5));
    [U_irf_RX16,S_irf_RX16,V_irf_RX16]               =
svd(svd_irf_RX16);
    Ue_irf_RX16                                      =
V_irf_RX16(1:NRFt_HB,1:Ns_HB);
    Te_irf_RX16                                      =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_irf_RX16                    = (Q_irf_RX16^(-
0.5))*Ue_irf_RX16*Te_irf_RX16;

    vrf_old25 = vrf_new25;
    vrf_new25                                        =
trace(vrf_irf_RX16*Vd_irf_RX16*Vd_irf_RX16'*vrf_
irf_RX16');
    tol25 = vrf_new25 - vrf_old25;

    jt25 = jt25 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO
```

```
Vt_irf_RX16        = vrf_irf_RX16*Vd_irf_RX16;
F2_irf_RX16                                    =
H_irf_RX16*Vt_irf_RX16*Vt_irf_RX16'*H_irf_RX16';

wrf_irf_RX16                                   =
ones(num_u_RX16,NRFr_HB);          %step 1
l2_irf_RX16        = length(num_u_RX16);
nij2_irf_RX16      = zeros(num_u_RX16,NRFr_HB);
Wrf_irf1_RX16      = zeros(num_u_RX16,NRFr_HB);
vrf_new26 = 100;
tol26 = 1;
jt26 = 0;

while (tol26 > 1e-5)
    for               j2_irf_RX16              =
1:1:NRFr_HB                                %step
2
        wrfj_irf_RX16           = wrf_irf_RX16;
        wrfj_irf_RX16(:,j2_irf_RX16)       = [];
        I2_irf_RX16               = eye(NRFr_HB-
1);   %step 3&4
        C2_irf_RX16              = I2_irf_RX16 +
(((1/num_u_RX16)/variance_sistem)*wrfj_irf_RX16'
*F2_irf_RX16*wrfj_irf_RX16);
        G2_irf_RX16                            =
(((1/num_u_RX16)/variance_sistem)*F2_irf_RX16)  -
(((1/num_u_RX16)^2/variance_sistem^2)*F2_irf_RX1
6*wrfj_irf_RX16*(C2_irf_RX16^(-
1))*wrfj_irf_RX16'*F2_irf_RX16);
        sigma2_irf_RX16 = 0;
        for i2_irf_RX16 = 1:1:num_u_RX16
            for L2_irf_RX16 = 1:1:num_u_RX16
                if L2_irf_RX16 ~= i2_irf_RX16
                sigma2_irf_RX16 = sigma2_irf_RX16
+     G2_irf_RX16(i2_irf_RX16,L2_irf_RX16)     *
wrf_irf_RX16(L2_irf_RX16,j2_irf_RX16);
                end
            end
            nij2_irf_RX16=sigma2_irf_RX16;
```

```
            if nij2_irf_RX16 == 0

wrf_irf_RX16(i2_irf_RX16,j2_irf_RX16) = 1;
            else

wrf_irf_RX16(i2_irf_RX16,j2_irf_RX16)           =
nij2_irf_RX16/abs(nij2_irf_RX16);
            end
            %i2_irf_RX16 = i2_irf + 1;
        end
      %j2_irf_RX16 = j2_irf + 1;
    end
    j_IRF_RX16                                  =
wrf_irf_RX16'*H_irf_RX16*vrf_irf_RX16*Vd_irf_RX1
6*Vd_irf_RX16'*vrf_irf_RX16'*H_irf_RX16'*wrf_irf
_RX16                                           +
variance_sistem*wrf_irf_RX16'*wrf_irf_RX16;
    J_IRF_RX16          = inv(j_IRF_RX16);
    Wd_irf_RX16                                 =
J_IRF_RX16*wrf_irf_RX16'*H_irf_RX16*vrf_irf_RX16
*Vd_irf_RX16;
    Wt_irf_RX16      = wrf_irf_RX16*Wd_irf_RX16;

    vrf_old26 = vrf_new26;
    vrf_new26                                   =
trace(wrf_irf_RX16*Wd_irf_RX16*Wd_irf_RX16'*wrf_
irf_RX16');
    tol26 = vrf_new26 - vrf_old26;

jt26 = jt26 + 1;
end

SE_IRF_RX16(SNRLoop,n_sistem)                   =
real(log2(det((eye(num_u_RX16))                 +
(1/variance_sistem)*Wt_irf_RX16*(inv(Wt_irf_RX16
'*Wt_irf_RX16))*Wt_irf_RX16'*H_irf_RX16*Vt_irf_R
X16*Vt_irf_RX16'*H_irf_RX16')));

F1_urlos_RX64        = H_urlos'*H_urlos;
gamma_urlos_RX64                                =
```

```
sqrt(P_sistem/(num_m_RX64*NRFt_HB));
vrf_urlos_RX64                                  =
ones(num_m_RX64,NRFt_HB);                % step 1
l1_urlos_RX64         = length(num_m_RX64);
nij_urlos_RX64        = zeros(num_m_RX64,NRFt_HB);
vrf_urlos1_RX64       = zeros(num_m_RX64,NRFt_HB);
vrf_new27          = 100;
tol27              = 1;
jt27               = 0;

while (tol27 > 1e-5)
    for              j1_urlos_RX64              =
1:1:NRFt_HB                              % step 2
        vrfj_urlos_RX64                         =
vrf_urlos_RX64;
        vrfj_urlos_RX64(:,j1_urlos_RX64)        =
[];
        I1_urlos_RX64              = eye(NRFt_HB-
1);
        C1_urlos_RX64             = I1_urlos_RX64
+
((gamma_urlos_RX64^2/variance_sistem)*vrfj_urlos
_RX64'*F1_urlos_RX64*vrfj_urlos_RX64);          %
step 3
        G1_urlos_RX64                           =
((gamma_urlos_RX64^2/variance_sistem)*F1_urlos_R
X64)                                            -
((gamma_urlos_RX64^4/variance_sistem^2)*F1_urlos
_RX64*vrfj_urlos_RX64*(C1_urlos_RX64^(-
1))*vrfj_urlos_RX64'*F1_urlos_RX64);   % step 4
        sigma_urlos_RX64 = 0;
        for            i1_urlos_RX64            =
1:1:num_m_RX64                           % step 5
            for L1_urlos_RX64 = 1:1:num_m_RX64
                if L1_urlos_RX64 ~= i1_urlos_RX64
                sigma_urlos_RX64                =
sigma_urlos_RX64                                +
G1_urlos_RX64(i1_urlos_RX64,L1_urlos_RX64)      *
vrf_urlos_RX64(L1_urlos_RX64,j1_urlos_RX64);
    % step 6
```

```
                end
            end
            nij_urlos_RX64 = sigma_urlos_RX64;
            if          nij_urlos_RX64          ==
0                         % step 7

vrf_urlos_RX64(i1_urlos_RX64,j1_urlos_RX64) = 1;
            else

vrf_urlos_RX64(i1_urlos_RX64,j1_urlos_RX64)     =
nij_urlos_RX64/abs(nij_urlos_RX64);
            end
        end
          cekv_FDB = abs(vrf_urlos_RX64);
    end
    I2_urlos_RX64                 = eye(NRFr_HB);
    VdVdh_urlos_RX64                                =
(sqrt(gamma_urlos_RX64^2))*I2_urlos_RX64;
    Heff_urlos_RX64                                 =
H_urlos*vrf_urlos_RX64;
    Q_urlos_RX64                                    =
vrf_urlos_RX64'*vrf_urlos_RX64;
    svd_urlos_RX64                                  =
Heff_urlos_RX64*(Q_urlos_RX64^(-0.5));
    [U_urlos_RX64,S_urlos_RX64,V_urlos_RX64]
= svd(svd_urlos_RX64);
    Ue_urlos_RX64                                   =
V_urlos_RX64(1:NRFt_HB,1:Ns_HB);
    Te_urlos_RX64                                   =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_urlos_RX64              = (Q_urlos_RX64^(-
0.5))*Ue_urlos_RX64*Te_urlos_RX64;

    vrf_old27 = vrf_new27;
    vrf_new27                                       =
trace(vrf_urlos_RX64*Vd_urlos_RX64*Vd_urlos_RX64
'*vrf_urlos_RX64');
    tol27 = vrf_new27 - vrf_old27;

    jt27 = jt27 + 1;
```

```
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_RX64                                    =
vrf_urlos_RX64*Vd_urlos_RX64;
F2_urlos_RX64                                    =
H_urlos*Vt_urlos_RX64*Vt_urlos_RX64'*H_urlos';

%P = SNR;
wrf_urlos_RX64                                   =
ones(num_u_RX64,NRFr_HB);          %step 1
l2_urlos_RX64         = length(num_u_RX64);
nij2_urlos_RX64       = zeros(num_u_RX64,NRFr_HB);
Wrf_urlos1_RX64       = zeros(num_u_RX64,NRFr_HB);
vrf_new28 = 100;
tol28 = 1;
jt28 = 0;

while (tol28 > 1e-5)
    for              j2_urlos_RX64               =
1:1:NRFr_HB                                  %step
2
        wrfj_urlos_RX64          = wrf_urlos_RX64;
        wrfj_urlos_RX64(:,j2_urlos_RX64)         =
[];
        I2_urlos_RX64               = eye(NRFr_HB-
1);   %step 3&4
        C2_urlos_RX64              = I2_urlos_RX64
+
(((1/num_u_RX64)/variance_sistem)*wrfj_urlos_RX6
4'*F2_urlos_RX64*wrfj_urlos_RX64);
        G2_urlos_RX64                            =
(((1/num_u_RX64)/variance_sistem)*F2_urlos_RX64)
-
(((1/num_u_RX64)^2/variance_sistem^2)*F2_urlos_R
X64*wrfj_urlos_RX64*(C2_urlos_RX64^(-
1))*wrfj_urlos_RX64'*F2_urlos_RX64);
        sigma2_urlos_RX64 = 0;
```

```
        for i2_urlos_RX64 = 1:1:num_u_RX64
            for L2_urlos_RX64 = 1:1:num_u_RX64
                if L2_urlos_RX64 ~= i2_urlos_RX64
                sigma2_urlos_RX64                =
sigma2_urlos_RX64                                +
G2_urlos_RX64(i2_urlos_RX64,L2_urlos_RX64)       *
wrf_urlos_RX64(L2_urlos_RX64,j2_urlos_RX64);
                end
            end
            nij2_urlos_RX64=sigma2_urlos_RX64;
            if nij2_urlos_RX64 == 0

wrf_urlos_RX64(i2_urlos_RX64,j2_urlos_RX64) = 1;
            else

wrf_urlos_RX64(i2_urlos_RX64,j2_urlos_RX64)      =
nij2_urlos_RX64/abs(nij2_urlos_RX64);
            end
            %i2_urlos_RX64 = i2_urlos + 1;
        end
      %j2_urlos_RX64 = j2_urlos + 1;
    end
    j_urlos_RX64                                 =
wrf_urlos_RX64'*H_urlos*vrf_urlos_RX64*Vd_urlos_
RX64*Vd_urlos_RX64'*vrf_urlos_RX64'*H_urlos'*wrf
_urlos_RX64                                      +
variance_sistem*wrf_urlos_RX64'*wrf_urlos_RX64;
    J_urlos_RX64           = inv(j_urlos_RX64);
    Wd_urlos_RX64                                =
J_urlos_RX64*wrf_urlos_RX64'*H_urlos*vrf_urlos_R
X64*Vd_urlos_RX64;
    Wt_urlos_RX64                                =
wrf_urlos_RX64*Wd_urlos_RX64;

    vrf_old28 = vrf_new28;
    vrf_new28                                    =
trace(wrf_urlos_RX64*Wd_urlos_RX64*Wd_urlos_RX64
'*wrf_urlos_RX64');
    tol28 = vrf_old28 - vrf_new28;
```

```
jt28 = jt28 + 1;
end

SE_urlos_RX64(SNRLoop,n_sistem)                    =
real(log2(det((eye(num_u_RX64))                    +
(1/variance_sistem)*Wt_urlos_RX64*(inv(Wt_urlos_
RX64'*Wt_urlos_RX64))*Wt_urlos_RX64'*H_urlos*Vt_
urlos_RX64*Vt_urlos_RX64'*H_urlos')));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%  SISTEM  HYBRID  BEAMFORMING  POINT-TO-POINT
MIMO dengan RX = 8

H_urlos_RX32          =  H_urlos(1:num_u_RX32,:);
F1_urlos_RX32                                      =
H_urlos_RX32(:,1:num_m_RX32)'*H_urlos_RX32(:,1:n
um_m_RX32);
gamma_urlos_RX32                                   =
sqrt(P_sistem/(num_m_RX32*NRFt_HB));
vrf_urlos_RX32                                     =
ones(num_m_RX32,NRFt_HB);                  % step 1
l1_urlos_RX32         = length(num_m_RX32);
nij_urlos_RX32        = zeros(num_m_RX32,NRFt_HB);
vrf_urlos1_RX32       = zeros(num_m_RX32,NRFt_HB);
vrf_new29          = 100;
tol29              = 1;
jt29               = 0;

while (tol29 > 1e-5)
    for              j1_urlos_RX32                 =
1:1:NRFt_HB                                % step 2
        vrfj_urlos_RX32                            =
vrf_urlos_RX32;
        vrfj_urlos_RX32(:,j1_urlos_RX32)           =
[];
        I1_urlos_RX32               = eye(NRFt_HB-
1);
        C1_urlos_RX32             = I1_urlos_RX32
+
```

```
((gamma_urlos_RX32^2/variance_sistem)*vrfj_urlos
_RX32'*F1_urlos_RX32*vrfj_urlos_RX32);          %
step 3
        G1_urlos_RX32                                =
((gamma_urlos_RX32^2/variance_sistem)*F1_urlos_R
X32)                                                 -
((gamma_urlos_RX32^4/variance_sistem^2)*F1_urlos
_RX32*vrfj_urlos_RX32*(C1_urlos_RX32^(-
1))*vrfj_urlos_RX32'*F1_urlos_RX32);    % step 4
        sigma_urlos_RX32 = 0;
        for            i1_urlos_RX32             =
1:1:num_m_RX32                          % step 5
            for L1_urlos_RX32 = 1:1:num_m_RX32
                if L1_urlos_RX32 ~= i1_urlos_RX32
                sigma_urlos_RX32                 =
sigma_urlos_RX32                                 +
G1_urlos_RX32(i1_urlos_RX32,L1_urlos_RX32)       *
vrf_urlos_RX32(L1_urlos_RX32,j1_urlos_RX32);
    % step 6
                end
            end
            nij_urlos_RX32=sigma_urlos_RX32;
            if          nij_urlos_RX32          ==
0                         % step 7

vrf_urlos_RX32(i1_urlos_RX32,j1_urlos_RX32) = 1;
            else

vrf_urlos_RX32(i1_urlos_RX32,j1_urlos_RX32)      =
nij_urlos_RX32/abs(nij_urlos_RX32);
            end
        end
          cekv_FDB = abs(vrf_urlos_RX32);
    end
    I2_urlos_RX32                  = eye(NRFr_HB);
    VdVdH_urlos_32_RX32                          =
(sqrt(gamma_urlos_RX32^2))*I2_urlos_RX32;
    Heff_urlos_RX32                              =
H_urlos_RX32*vrf_urlos_RX32;
    Q_urlos_RX32                                 =
```

```
vrf_urlos_RX32'*vrf_urlos_RX32;
    svd_urlos_RX32                                  =
Heff_urlos_RX32*(Q_urlos_RX32^(-0.5));
    [U_urlos_RX32,S_urlos_RX32,V_urlos_RX32]
= svd(svd_urlos_RX32);
    Ue_urlos_RX32                                   =
V_urlos_RX32(1:NRFt_HB,1:Ns_HB);
    Te_urlos_RX32                                   =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_urlos_RX32              = (Q_urlos_RX32^(-
0.5))*Ue_urlos_RX32*Te_urlos_RX32;

    vrf_old29 = vrf_new29;
    vrf_new29                                       =
trace(vrf_urlos_RX32*Vd_urlos_RX32*Vd_urlos_RX32
'*vrf_urlos_RX32');
    tol29 = vrf_new29 - vrf_old29;

    jt29 = jt29 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_RX32                                       =
vrf_urlos_RX32*Vd_urlos_RX32;
F2_urlos_RX32                                       =
H_urlos_RX32*Vt_urlos_RX32*Vt_urlos_RX32'*H_urlo
s_RX32';

%P = SNR;
wrf_urlos_RX32                                      =
ones(num_u_RX32,NRFr_HB);          %step 1
l2_urlos_RX32         = length(num_u_RX32);
nij2_urlos_RX32       = zeros(num_u_RX32,NRFr_HB);
Wrf_urlos1_RX32       = zeros(num_u_RX32,NRFr_HB);
vrf_new30 = 100;
tol30 = 1;
jt30 = 0;
```

```
while (tol30 > 1e-5)
    for              j2_urlos_RX32              =
1:1:NRFr_HB                                    %step
2
        wrfj_urlos_RX32         = wrf_urlos_RX32;
        wrfj_urlos_RX32(:,j2_urlos_RX32)        =
[];
        I2_urlos_RX32              = eye(NRFr_HB-
1);   %step 3&4
        C2_urlos_RX32             = I2_urlos_RX32
+
(((1/num_u_RX32)/variance_sistem)*wrfj_urlos_RX3
2'*F2_urlos_RX32*wrfj_urlos_RX32);
        G2_urlos_RX32                           =
(((1/num_u_RX32)/variance_sistem)*F2_urlos_RX32)
-
(((1/num_u_RX32)^2/variance_sistem^2)*F2_urlos_R
X32*wrfj_urlos_RX32*(C2_urlos_RX32^(-
1))*wrfj_urlos_RX32'*F2_urlos_RX32);
        sigma2_urlos_RX32 = 0;
        for i2_urlos_RX32 = 1:1:num_u_RX32
            for L2_urlos_RX32 = 1:1:num_u_RX32
                if L2_urlos_RX32 ~= i2_urlos_RX32
                sigma2_urlos_RX32               =
sigma2_urlos_RX32                               +
G2_urlos_RX32(i2_urlos_RX32,L2_urlos_RX32)      *
wrf_urlos_RX32(L2_urlos_RX32,j2_urlos_RX32);
                end
            end
            nij2_urlos_RX32=sigma2_urlos_RX32;
            if nij2_urlos_RX32 == 0

wrf_urlos_RX32(i2_urlos_RX32,j2_urlos_RX32) = 1;
            else

wrf_urlos_RX32(i2_urlos_RX32,j2_urlos_RX32)     =
nij2_urlos_RX32/abs(nij2_urlos_RX32);
            end
            %i2_urlos_RX32 = i2_urlos + 1;
        end
```

```
      %j2_urlos_RX32 = j2_urlos + 1;
    end
    j_urlos_RX32                                    =
wrf_urlos_RX32'*H_urlos_RX32*vrf_urlos_RX32*Vd_u
rlos_RX32*Vd_urlos_RX32'*vrf_urlos_RX32'*H_urlos
_RX32'*wrf_urlos_RX32                               +
variance_sistem*wrf_urlos_RX32'*wrf_urlos_RX32;
    J_urlos_RX32           = inv(j_urlos_RX32);
    Wd_urlos_RX32                                   =
J_urlos_RX32*wrf_urlos_RX32'*H_urlos_RX32*vrf_ur
los_RX32*Vd_urlos_RX32;
    Wt_urlos_RX32                                   =
wrf_urlos_RX32*Wd_urlos_RX32;

    vrf_old30 = vrf_new30;
    vrf_new30                                       =
trace(wrf_urlos_RX32*Wd_urlos_RX32*Wd_urlos_RX32
'*wrf_urlos_RX32');
    tol30 = vrf_old30 - vrf_new30;

jt30 = jt30 + 1;
end

SE_urlos_RX32(SNRLoop,n_sistem)                     =
real(log2(det((eye(num_u_RX32))                     +
(1/variance_sistem)*Wt_urlos_RX32*(inv(Wt_urlos_
RX32'*Wt_urlos_RX32))*Wt_urlos_RX32'*H_urlos_RX3
2*Vt_urlos_RX32*Vt_urlos_RX32'*H_urlos_RX32')));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%  SISTEM  HYBRID  BEAMFORMING  POINT-TO-POINT
MIMO dengan RX = 8

H_urlos_RX16          =  H_urlos(1:num_u_RX16,:);
F1_urlos_RX16                                       =
H_urlos_RX16(:,1:num_m_RX16)'*H_urlos_RX16(:,1:n
um_m_RX16);
gamma_urlos_RX16                                    =
sqrt(P_sistem/(num_m_RX16*NRFt_HB));
```

```
vrf_urlos_RX16                                          =
ones(num_m_RX16,NRFt_HB);                % step 1
l1_urlos_RX16          = length(num_m_RX16);
nij_urlos_RX16         = zeros(num_m_RX16,NRFt_HB);
vrf_urlos1_RX16        = zeros(num_m_RX16,NRFt_HB);
vrf_new31          = 100;
tol31              = 1;
jt31               = 0;

while (tol31 > 1e-5)
    for              j1_urlos_RX16                  =
1:1:NRFt_HB                              % step 2
        vrfj_urlos_RX16                            =
vrf_urlos_RX16;
        vrfj_urlos_RX16(:,j1_urlos_RX16)           =
[];
        I1_urlos_RX16              = eye(NRFt_HB-
1);
        C1_urlos_RX16             = I1_urlos_RX16
+
((gamma_urlos_RX16^2/variance_sistem)*vrfj_urlos
_RX16'*F1_urlos_RX16*vrfj_urlos_RX16);          %
step 3
        G1_urlos_RX16                              =
((gamma_urlos_RX16^2/variance_sistem)*F1_urlos_R
X16)                                               -
((gamma_urlos_RX16^4/variance_sistem^2)*F1_urlos
_RX16*vrfj_urlos_RX16*(C1_urlos_RX16^(-
1))*vrfj_urlos_RX16'*F1_urlos_RX16);    % step 4
        sigma_urlos_RX16 = 0;
        for            i1_urlos_RX16               =
1:1:num_m_RX16                           % step 5
            for L1_urlos_RX16 = 1:1:num_m_RX16
                if L1_urlos_RX16 ~= i1_urlos_RX16
                sigma_urlos_RX16                   =
sigma_urlos_RX16                                   +
G1_urlos_RX16(i1_urlos_RX16,L1_urlos_RX16)         *
vrf_urlos_RX16(L1_urlos_RX16,j1_urlos_RX16);
    % step 6
                end
```

```
            end
            nij_urlos_RX16=sigma_urlos_RX16;
            if          nij_urlos_RX16          ==
0                         % step 7

vrf_urlos_RX16(i1_urlos_RX16,j1_urlos_RX16) = 1;
            else

vrf_urlos_RX16(i1_urlos_RX16,j1_urlos_RX16)     =
nij_urlos_RX16/abs(nij_urlos_RX16);
            end
         end
           cekv_FDB = abs(vrf_urlos_RX16);
    end
    I2_urlos_RX16                 = eye(NRFr_HB);
    VdVdH_urlos_16_RX16                           =
(sqrt(gamma_urlos_RX16^2))*I2_urlos_RX16;
    Heff_urlos_RX16                               =
H_urlos_RX16*vrf_urlos_RX16;
    Q_urlos_RX16                                  =
vrf_urlos_RX16'*vrf_urlos_RX16;
    svd_urlos_RX16                                =
Heff_urlos_RX16*(Q_urlos_RX16^(-0.5));
    [U_urlos_RX16,S_urlos_RX16,V_urlos_RX16]
= svd(svd_urlos_RX16);
    Ue_urlos_RX16                                 =
V_urlos_RX16(1:NRFt_HB,1:Ns_HB);
    Te_urlos_RX16                                 =
sqrt(P_sistem/NRFt_HB)*(eye(Ns_HB));
    Vd_urlos_RX16              = (Q_urlos_RX16^(-
0.5))*Ue_urlos_RX16*Te_urlos_RX16;

    vrf_old31 = vrf_new31;
    vrf_new31                                     =
trace(vrf_urlos_RX16*Vd_urlos_RX16*Vd_urlos_RX16
'*vrf_urlos_RX16');
    tol31 = vrf_new31 - vrf_old31;

    jt31 = jt31 + 1;
end
```

```
% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_RX16                                    =
vrf_urlos_RX16*Vd_urlos_RX16;
F2_urlos_RX16                                    =
H_urlos_RX16*Vt_urlos_RX16*Vt_urlos_RX16'*H_urlo
s_RX16';

%P = SNR;
wrf_urlos_RX16                                   =
ones(num_u_RX16,NRFr_HB);          %step 1
l2_urlos_RX16        = length(num_u_RX16);
nij2_urlos_RX16      = zeros(num_u_RX16,NRFr_HB);
Wrf_urlos1_RX16      = zeros(num_u_RX16,NRFr_HB);
vrf_new32 = 100;
tol32 = 1;
jt32 = 0;

while (tol32 > 1e-5)
    for              j2_urlos_RX16               =
1:1:NRFr_HB                                  %step
2
        wrfj_urlos_RX16         = wrf_urlos_RX16;
        wrfj_urlos_RX16(:,j2_urlos_RX16)         =
[];
        I2_urlos_RX16              = eye(NRFr_HB-
1);   %step 3&4
        C2_urlos_RX16             = I2_urlos_RX16
+
(((1/num_u_RX16)/variance_sistem)*wrfj_urlos_RX1
6'*F2_urlos_RX16*wrfj_urlos_RX16);
        G2_urlos_RX16                            =
(((1/num_u_RX16)/variance_sistem)*F2_urlos_RX16)
-
(((1/num_u_RX16)^2/variance_sistem^2)*F2_urlos_R
X16*wrfj_urlos_RX16*(C2_urlos_RX16^(-
1))*wrfj_urlos_RX16'*F2_urlos_RX16);
        sigma2_urlos_RX16 = 0;
```

```
        for i2_urlos_RX16 = 1:1:num_u_RX16
            for L2_urlos_RX16 = 1:1:num_u_RX16
                if L2_urlos_RX16 ~= i2_urlos_RX16
                sigma2_urlos_RX16                =
sigma2_urlos_RX16                                +
G2_urlos_RX16(i2_urlos_RX16,L2_urlos_RX16)       *
wrf_urlos_RX16(L2_urlos_RX16,j2_urlos_RX16);
                end
            end
            nij2_urlos_RX16=sigma2_urlos_RX16;
            if nij2_urlos_RX16 == 0

wrf_urlos_RX16(i2_urlos_RX16,j2_urlos_RX16) = 1;
            else

wrf_urlos_RX16(i2_urlos_RX16,j2_urlos_RX16)      =
nij2_urlos_RX16/abs(nij2_urlos_RX16);
            end
            %i2_urlos_RX16 = i2_urlos + 1;
        end
      %j2_urlos_RX16 = j2_urlos + 1;
    end
    j_urlos_RX16                                 =
wrf_urlos_RX16'*H_urlos_RX16*vrf_urlos_RX16*Vd_u
rlos_RX16*Vd_urlos_RX16'*vrf_urlos_RX16'*H_urlos
_RX16'*wrf_urlos_RX16                            +
variance_sistem*wrf_urlos_RX16'*wrf_urlos_RX16;
    J_urlos_RX16           = inv(j_urlos_RX16);
    Wd_urlos_RX16                                =
J_urlos_RX16*wrf_urlos_RX16'*H_urlos_RX16*vrf_ur
los_RX16*Vd_urlos_RX16;
    Wt_urlos_RX16                                =
wrf_urlos_RX16*Wd_urlos_RX16;

    vrf_old32 = vrf_new32;
    vrf_new32                                    =
trace(wrf_urlos_RX16*Wd_urlos_RX16*Wd_urlos_RX16
'*wrf_urlos_RX16');
    tol32 = vrf_new32 - vrf_old32;
```

```
jt32 = jt32 + 1;
end

SE_urlos_RX16(SNRLoop,n_sistem)                    =
real(log2(det((eye(num_u_RX16))                    +
(1/variance_sistem)*Wt_urlos_RX16*(inv(Wt_urlos_
RX16'*Wt_urlos_RX16))*Wt_urlos_RX16'*H_urlos_RX1
6*Vt_urlos_RX16*Vt_urlos_RX16'*H_urlos_RX16')));

%%%%%% MEMBUAT SISTEM DENGAN RF CHAIN = 4 pada
kanal IRF
F1_irf_RF4      = H_irf'*H_irf;
gamma_irf_RF4                                      =
sqrt(P_sistem/(num_m*NRFt_RF4));
vrf_irf_RF4     = ones(num_m,NRFt_RF4); % step 1
l1_irf_RF4      = length(num_m);
nij_irf_RF4     = zeros(num_m,NRFt_RF4);
vrf_irf1_RF4    = zeros(num_m,NRFt_RF4);
vrf_new33        = 100;
tol33            = 1;
jt33             = 0;

while (tol33 > 1e-5)
    for j1_irf_RF4      = 1:1:NRFt_RF4 % step 2
        vrfj_irf_RF4    = vrf_irf_RF4;
        vrfj_irf_RF4(:,j1_irf_RF4) = [];
        I1_irf_RF4  = eye(NRFt_RF4-1);
        C1_irf_RF4          =     I1_irf_RF4     +
((gamma_irf_RF4^2/variance_sistem)*vrfj_irf_RF4'
*F1_irf_RF4*vrfj_irf_RF4); % step 3
        G1_irf_RF4                                 =
((gamma_irf_RF4^2/variance_sistem)*F1_irf_RF4)  -
((gamma_irf_RF4^4/variance_sistem^2)*F1_irf_RF4*
vrfj_irf_RF4*(C1_irf_RF4^(-
1))*vrfj_irf_RF4'*F1_irf_RF4); % step 4
        sigma_irf_RF4 = 0;
        for i1_irf_RF4 = 1:1:num_m % step 5
            for L1_irf_RF4 = 1:1:num_m
                if L1_irf_RF4 ~= i1_irf_RF4
                    sigma_irf_RF4 = sigma_irf_RF4
```

```
+       G1_irf_RF4(i1_irf_RF4,L1_irf_RF4)       *
vrf_irf_RF4(L1_irf_RF4,j1_irf_RF4); % step 6
                end
            end
            nij_irf_RF4 = sigma_irf_RF4;
            if nij_irf_RF4 == 0 % step 7

vrf_irf_RF4(i1_irf_RF4,j1_irf_RF4) = 1;
            else

vrf_irf_RF4(i1_irf_RF4,j1_irf_RF4)              =
nij_irf_RF4/abs(nij_irf_RF4);
            end
        end
        cekv_irf_RF2 = abs(vrf_irf_RF4);
    end
    I2_irf_RF4      = eye(NRFr_RF4);
    VdVdh_irf_RF4                               =
(sqrt(gamma_irf_RF4^2))*I2_irf_RF4;
    Heff_irf_RF4    = H_irf*vrf_irf_RF4;
    Q_irf_RF4       = vrf_irf_RF4'*vrf_irf_RF4;
    svd_irf_RF4      = Heff_irf_RF4*(Q_irf_RF4^(-
0.5));
    [U_irf_RF4,S_irf_RF4,V_irf_RF4]             =
svd(svd_irf_RF4);
    Ue_irf_RF4                                  =
V_irf_RF4(1:Ns_RF4,1:Ns_RF4);
    Te_irf_RF4                                  =
sqrt(P_sistem/NRFt_RF4)*(eye(Ns_RF4));
    Vd_irf_RF4                  =   (Q_irf_RF4^(-
0.5))*Ue_irf_RF4*Te_irf_RF4;

    vrf_old33 = vrf_new33;
    vrf_new33                                   =
trace(vrf_irf_RF4*Vd_irf_RF4*Vd_irf_RF4'*vrf_irf
_RF4');
    tol33 = vrf_new33 - vrf_old33;
    jt33 = jt33 + 1;
end
```

```
% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO
Vt_irf_RF4 = vrf_irf_RF4*Vd_irf_RF4;
F2_irf_RF4 = H_irf*Vt_irf_RF4*Vt_irf_RF4'*H_irf';

wrf_irf_RF4     = ones(num_u,NRFr_RF4); %step 1
l2_irf_RF4      = length(num_u);
nij2_irf_RF4    = zeros(num_u,NRFr_RF4);
Wrf_irf1_RF4    = zeros(num_u,NRFr_RF4);
vrf_new34        = 100;
tol34            = 1;
jt34             = 0;

while (tol34 > 1e-5)
    for j2_irf_RF4 = 1:1:NRFr_RF4 %step 2
        wrfj_irf_RF4 = wrf_irf_RF4;
        wrfj_irf_RF4(:,j2_irf_RF4) = [];
        I2_irf_RF4 = eye(NRFr_RF4-1); %step 3&4
        C2_irf_RF4      =      I2_irf_RF4      +
(((1/num_u)/variance_sistem)*wrfj_irf_RF4'*F2_ir
f_RF4*wrfj_irf_RF4);
        G2_irf_RF4                              =
(((1/num_u)/variance_sistem)*F2_irf_RF4)        -
(((1/num_u)^2/variance_sistem^2)*F2_irf_RF4*wrfj
_irf_RF4*(C2_irf_RF4^(-
1))*wrfj_irf_RF4'*F2_irf_RF4);
        sigma2_irf_RF4 = 0;
        for i2_irf_RF4 = 1:1:num_u
            for L2_irf_RF4 = 1:1:num_u
                if L2_irf_RF4 ~= i2_irf_RF4
                    sigma2_irf_RF4              =
sigma2_irf_RF4                                  +
G2_irf_RF4(i2_irf_RF4,L2_irf_RF4)               *
wrf_irf_RF4(L2_irf_RF4,j2_irf_RF4);
                end
            end
            nij2_irf_RF4=sigma2_irf_RF4;
            if nij2_irf_RF4 == 0

wrf_irf_RF4(i2_irf_RF4,j2_irf_RF4) = 1;
```

```
            else

wrf_irf_RF4(i2_irf_RF4,j2_irf_RF4)              =
nij2_irf_RF4/abs(nij2_irf_RF4);
            end
            %i2_irf_RF4 = i2_irf + 1;
        end
        %j2_irf_RF4 = j2_irf + 1;
    end
    j_IRF_RF4                                   =
wrf_irf_RF4'*H_irf*vrf_irf_RF4*Vd_irf_RF4*Vd_irf
_RF4'*vrf_irf_RF4'*H_irf'*wrf_irf_RF4           +
variance_sistem*wrf_irf_RF4'*wrf_irf_RF4;
    J_IRF_RF4 = inv(j_IRF_RF4);
    Wd_irf_RF4                                  =
J_IRF_RF4*wrf_irf_RF4'*H_irf*vrf_irf_RF4*Vd_irf_
RF4;
    Wt_irf_RF4 = wrf_irf_RF4*Wd_irf_RF4;

    vrf_old34 = vrf_new34;
    vrf_new34                                   =
trace(wrf_irf_RF4*Wd_irf_RF4*Wd_irf_RF4'*wrf_irf
_RF4');
    tol34 = vrf_new34 - vrf_old34;

    jt34 = jt34 + 1;
end

SE_IRF_RF4(SNRLoop,n_sistem)                    =
real(log2(det((eye(num_u))                      +
(1/variance_sistem)*Wt_irf_RF4*(inv(Wt_irf_RF4'*
Wt_irf_RF4))*Wt_irf_RF4'*H_irf*Vt_irf_RF4*Vt_irf
_RF4'*H_irf')));

%%%%% MEMBUAT SISTEM DENGAN RF CHAIN = 2 pada
kanal IRF
F1_irf_RF2      = H_irf'*H_irf;
gamma_irf_RF2                                   =
sqrt(P_sistem/(num_m*NRFt_RF2));
vrf_irf_RF2     = ones(num_m,NRFt_RF2); % step 1
```

```
l1_irf_RF2      = length(num_m);
nij_irf_RF2     = zeros(num_m,NRFt_RF2);
vrf_irf1_RF2    = zeros(num_m,NRFt_RF2);
vrf_new35        = 100;
tol35            = 1;
jt35             = 0;

while (tol35 > 1e-5)
    for j1_irf_RF2 = 1:1:NRFt_RF2 % step 2
        vrfj_irf_RF2 = vrf_irf_RF2;
        vrfj_irf_RF2(:,j1_irf_RF2) = [];
        I1_irf_RF2 = eye(NRFt_RF2-1);
        C1_irf_RF2      =      I1_irf_RF2      +
((gamma_irf_RF2^2/variance_sistem)*vrfj_irf_RF2'
*F1_irf_RF2*vrfj_irf_RF2); % step 3
        G1_irf_RF2                              =
((gamma_irf_RF2^2/variance_sistem)*F1_irf_RF2)  -
((gamma_irf_RF2^4/variance_sistem^2)*F1_irf_RF2*
vrfj_irf_RF2*(C1_irf_RF2^(-
1))*vrfj_irf_RF2'*F1_irf_RF2); % step 4
        sigma_irf_RF2 = 0;
        for i1_irf_RF2 = 1:1:num_m % step 5
            for L1_irf_RF2 = 1:1:num_m
                if L1_irf_RF2 ~= i1_irf_RF2
                    sigma_irf_RF2 = sigma_irf_RF2
+       G1_irf_RF2(i1_irf_RF2,L1_irf_RF2)       *
vrf_irf_RF2(L1_irf_RF2,j1_irf_RF2); % step 6
                end
            end
            nij_irf_RF2=sigma_irf_RF2;
            if nij_irf_RF2 == 0 % step 7

vrf_irf_RF2(i1_irf_RF2,j1_irf_RF2) = 1;
            else

vrf_irf_RF2(i1_irf_RF2,j1_irf_RF2)              =
nij_irf_RF2/abs(nij_irf_RF2);
            end
        end
        cekv_irf_RF2 = abs(vrf_irf_RF2);
```

```
    end
    I2_irf_RF2      = eye(NRFr_RF2);
    VdVdh_irf_RF2                                   =
(sqrt(gamma_irf_RF2^2))*I2_irf_RF2;
    Heff_irf_RF2    = H_irf*vrf_irf_RF2;
    Q_irf_RF2       = vrf_irf_RF2'*vrf_irf_RF2;
    svd_irf_RF2      = Heff_irf_RF2*(Q_irf_RF2^(-
0.5));
    [U_irf_RF2,S_irf_RF2,V_irf_RF2]                 =
svd(svd_irf_RF2);
    Ue_irf_RF2                                      =
V_irf_RF2(1:Ns_RF2,1:Ns_RF2);
    Te_irf_RF2                                      =
sqrt(P_sistem/NRFt_RF2)*(eye(Ns_RF2));
    Vd_irf_RF2                  =   (Q_irf_RF2^(-
0.5))*Ue_irf_RF2*Te_irf_RF2;

    vrf_old35 = vrf_new35;
    vrf_new35                                       =
trace(vrf_irf_RF2*Vd_irf_RF2*Vd_irf_RF2'*vrf_irf
_RF2');
    tol35 = vrf_new35 - vrf_old35;

    jt35 = jt35 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_RF2 = vrf_irf_RF2*Vd_irf_RF2;
F2_irf_RF2 = H_irf*Vt_irf_RF2*Vt_irf_RF2'*H_irf';

%P = SNR;
wrf_irf_RF2     = ones(num_u,NRFr_RF2); %step 1
l2_irf_RF2      = length(num_u);
nij2_irf_RF2    = zeros(num_u,NRFr_RF2);
Wrf_irf1_RF2    = zeros(num_u,NRFr_RF2);
vrf_new36        = 100;
tol36            = 1;
jt36             = 0;
```

```
while (tol36 > 1e-5)
    for j2_irf_RF2 = 1:1:NRFr_RF2 %step 2
        wrfj_irf_RF2 = wrf_irf_RF2;
        wrfj_irf_RF2(:,j2_irf_RF2) = [];
        I2_irf_RF2 = eye(NRFr_RF2-1); %step 3&4
        C2_irf_RF2      =      I2_irf_RF2      +
(((1/num_u)/variance_sistem)*wrfj_irf_RF2'*F2_ir
f_RF2*wrfj_irf_RF2);
        G2_irf_RF2                                =
(((1/num_u)/variance_sistem)*F2_irf_RF2)        -
(((1/num_u)^2/variance_sistem^2)*F2_irf_RF2*wrfj
_irf_RF2*(C2_irf_RF2^(-
1))*wrfj_irf_RF2'*F2_irf_RF2);
        sigma2_irf_RF2 = 0;
        for i2_irf_RF2 = 1:1:num_u
            for L2_irf_RF2 = 1:1:num_u
                if L2_irf_RF2 ~= i2_irf_RF2
                    sigma2_irf_RF2              =
sigma2_irf_RF2                                  +
G2_irf_RF2(i2_irf_RF2,L2_irf_RF2)               *
wrf_irf_RF2(L2_irf_RF2,j2_irf_RF2);
                end
            end
            nij2_irf_RF2=sigma2_irf_RF2;
            if nij2_irf_RF2 == 0

wrf_irf_RF2(i2_irf_RF2,j2_irf_RF2) = 1;
            else

wrf_irf_RF2(i2_irf_RF2,j2_irf_RF2)              =
nij2_irf_RF2/abs(nij2_irf_RF2);
            end
            %i2_irf_RF2 = i2_irf + 1;
        end
        %j2_irf_RF2 = j2_irf + 1;
    end
    j_IRF_RF2                                   =
wrf_irf_RF2'*H_irf*vrf_irf_RF2*Vd_irf_RF2*Vd_irf
_RF2'*vrf_irf_RF2'*H_irf'*wrf_irf_RF2           +
```

```
variance_sistem*wrf_irf_RF2'*wrf_irf_RF2;
    J_IRF_RF2 = inv(j_IRF_RF2);
    Wd_irf_RF2                                    =
J_IRF_RF2*wrf_irf_RF2'*H_irf*vrf_irf_RF2*Vd_irf_
RF2;
    Wt_irf_RF2 = wrf_irf_RF2*Wd_irf_RF2;

    vrf_old36 = vrf_new36;
    vrf_new36                                     =
trace(wrf_irf_RF2*Wd_irf_RF2*Wd_irf_RF2'*wrf_irf
_RF2');
    tol36 = vrf_new36 - vrf_old36;

    jt36 = jt36 + 1;
end

SE_IRF_RF2(SNRLoop,n_sistem)                      =
real(log2(det((eye(num_u))                        +
(1/variance_sistem)*Wt_irf_RF2*(inv(Wt_irf_RF2'*
Wt_irf_RF2))*Wt_irf_RF2'*H_irf*Vt_irf_RF2*Vt_irf
_RF2'*H_irf')));

%%%%%% MEMBUAT SISTEM DENGAN RF CHAIN = 4 pada
kanal URLOS
F1_urlos_RF4    = H_urlos'*H_urlos;
gamma_urlos_RF4                                   =
sqrt(P_sistem/(num_m*NRFt_RF4));
vrf_urlos_RF4   = ones(num_m,NRFt_RF4); % step 1
l1_urlos_RF4    = length(num_m);
nij_urlos_RF4   = zeros(num_m,NRFt_RF4);
vrf_urlos1_RF4  = zeros(num_m,NRFt_RF4);
vrf_new37        = 100;
tol37            = 1;
jt37             = 0;

while (tol37 > 1e-5)
    for j1_urlos_RF4 = 1:1:NRFt_RF4 % step 2
        vrfj_urlos_RF4 = vrf_urlos_RF4;
        vrfj_urlos_RF4(:,j1_urlos_RF4) = [];
        I1_urlos_RF4 = eye(NRFt_RF4-1);
```

```
        C1_urlos_RF4     =     I1_urlos_RF4     +
((gamma_urlos_RF4^2/variance_sistem)*vrfj_urlos_
RF4'*F1_urlos_RF4*vrfj_urlos_RF4); % step 3
        G1_urlos_RF4                             =
((gamma_urlos_RF4^2/variance_sistem)*F1_urlos_RF
4)                                               -
((gamma_urlos_RF4^4/variance_sistem^2)*F1_urlos_
RF4*vrfj_urlos_RF4*(C1_urlos_RF4^(-
1))*vrfj_urlos_RF4'*F1_urlos_RF4); % step 4
        sigma_urlos_RF4 = 0;
        for i1_urlos_RF4 = 1:1:num_m % step 5
            for L1_urlos_RF4 = 1:1:num_m
                if L1_urlos_RF4 ~= i1_urlos_RF4
                    sigma_urlos_RF4              =
sigma_urlos_RF4                                  +
G1_urlos_RF4(i1_urlos_RF4,L1_urlos_RF4)          *
vrf_urlos_RF4(L1_urlos_RF4,j1_urlos_RF4); % step
6
                end
            end
            nij_urlos_RF4=sigma_urlos_RF4;
            if nij_urlos_RF4 == 0 % step 7

vrf_urlos_RF4(i1_urlos_RF4,j1_urlos_RF4) = 1;
            else

vrf_urlos_RF4(i1_urlos_RF4,j1_urlos_RF4)         =
nij_urlos_RF4/abs(nij_urlos_RF4);
            end
        end
        cekv_RF2 = abs(vrf_urlos_RF4);
    end
    I2_urlos_RF4 = eye(NRFr_RF4);
    VdVdh_urlos_RF4                              =
(sqrt(gamma_urlos_RF4^2))*I2_urlos_RF4;
    Heff_urlos_RF4 = H_urlos*vrf_urlos_RF4;
    Q_urlos_RF4 = vrf_urlos_RF4'*vrf_urlos_RF4;
    svd_urlos_RF4                                =
Heff_urlos_RF4*(Q_urlos_RF4^(-0.5));
    [U_urlos_RF4,S_urlos_RF4,V_urlos_RF4]        =
```

```
svd(svd_urlos_RF4);
    Ue_urlos_RF4                                    =
V_urlos_RF4(1:Ns_RF4,1:Ns_RF4);
    Te_urlos_RF4                                    =
sqrt(P_sistem/NRFt_RF4)*(eye(Ns_RF4));
    Vd_urlos_RF4          =          (Q_urlos_RF4^(-
0.5))*Ue_urlos_RF4*Te_urlos_RF4;

    vrf_old37 = vrf_new37;
    vrf_new37                                       =
trace(vrf_urlos_RF4*Vd_urlos_RF4*Vd_urlos_RF4'*v
rf_urlos_RF4');
    tol37 = vrf_new37 - vrf_old37;

    jt37 = jt37 + 1;
end
% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_RF4 = vrf_urlos_RF4*Vd_urlos_RF4;
F2_urlos_RF4                                        =
H_urlos*Vt_urlos_RF4*Vt_urlos_RF4'*H_urlos';

%P = SNR;
wrf_urlos_RF4    = ones(num_u,NRFr_RF4); %step 1
l2_urlos_RF4     = length(num_u);
nij2_urlos_RF4   = zeros(num_u,NRFr_RF4);
Wrf_urlos1_RF4   = zeros(num_u,NRFr_RF4);
vrf_new38         = 100;
tol38             = 1;
jt38              = 0;

while (tol38 > 1e-5)
    for j2_urlos_RF4 = 1:1:NRFr_RF4 %step 2
        wrfj_urlos_RF4 = wrf_urlos_RF4;
        wrfj_urlos_RF4(:,j2_urlos_RF4) = [];
        I2_urlos_RF4 = eye(NRFr_RF4-1); %step 3&4
        C2_urlos_RF4     =     I2_urlos_RF4     +
(((1/num_u)/variance_sistem)*wrfj_urlos_RF4'*F2_
urlos_RF4*wrfj_urlos_RF4);
```

```
        G2_urlos_RF4                                    =
(((1/num_u)/variance_sistem)*F2_urlos_RF4)      -
(((1/num_u)^2/variance_sistem^2)*F2_urlos_RF4*wr
fj_urlos_RF4*(C2_urlos_RF4^(-
1))*wrfj_urlos_RF4'*F2_urlos_RF4);
        sigma2_urlos_RF4 = 0;
        for i2_urlos_RF4 = 1:1:num_u
            for L2_urlos_RF4 = 1:1:num_u
                if L2_urlos_RF4 ~= i2_urlos_RF4
                    sigma2_urlos_RF4            =
sigma2_urlos_RF4                                +
G2_urlos_RF4(i2_urlos_RF4,L2_urlos_RF4)         *
wrf_urlos_RF4(L2_urlos_RF4,j2_urlos_RF4);
                end
            end
            nij2_urlos_RF4=sigma2_urlos_RF4;
            if nij2_urlos_RF4 == 0

wrf_urlos_RF4(i2_urlos_RF4,j2_urlos_RF4) = 1;
            else

wrf_urlos_RF4(i2_urlos_RF4,j2_urlos_RF4)        =
nij2_urlos_RF4/abs(nij2_urlos_RF4);
            end
            %i2_urlos_RF4 = i2_urlos + 1;
        end
        %j2_urlos_RF4 = j2_urlos + 1;
    end
    j_URLOS_RF4                                     =
wrf_urlos_RF4'*H_urlos*vrf_urlos_RF4*Vd_urlos_RF
4*Vd_urlos_RF4'*vrf_urlos_RF4'*H_urlos'*wrf_urlo
s_RF4                                               +
variance_sistem*wrf_urlos_RF4'*wrf_urlos_RF4;
    J_URLOS_RF4 = inv(j_URLOS_RF4);
    Wd_urlos_RF4                                    =
J_URLOS_RF4*wrf_urlos_RF4'*H_urlos*vrf_urlos_RF4
*Vd_urlos_RF4;
    Wt_urlos_RF4 = wrf_urlos_RF4*Wd_urlos_RF4;

    vrf_old38 = vrf_new38;
```

```
    vrf_new38                                          =
trace(wrf_urlos_RF4*Wd_urlos_RF4*Wd_urlos_RF4'*w
rf_urlos_RF4');
    tol38 = vrf_new38 - vrf_old38;

    jt38 = jt38 + 1;
end

SE_URLOS_RF4(SNRLoop,n_sistem)                     =
real(log2(det((eye(num_u))                         +
(1/variance_sistem)*Wt_urlos_RF4*(inv(Wt_urlos_R
F4'*Wt_urlos_RF4))*Wt_urlos_RF4'*H_urlos*Vt_urlo
s_RF4*Vt_urlos_RF4'*H_urlos')));

%%%%%% MEMBUAT SISTEM DENGAN RF CHAIN = 2 pada
kanal URLOS
F1_urlos_RF2    = H_urlos'*H_urlos;
gamma_urlos_RF2                                    =
sqrt(P_sistem/(num_m*NRFt_RF2));
vrf_urlos_RF2   = ones(num_m,NRFt_RF2); % step 1
l1_urlos_RF2    = length(num_m);
nij_urlos_RF2   = zeros(num_m,NRFt_RF2);
vrf_urlos1_RF2  = zeros(num_m,NRFt_RF2);
vrf_new39        = 100;
tol39            = 1;
jt39             = 0;

while (tol39 > 1e-5)
    for j1_urlos_RF2 = 1:1:NRFt_RF2 % step 2
        vrfj_urlos_RF2 = vrf_urlos_RF2;
        vrfj_urlos_RF2(:,j1_urlos_RF2) = [];
        I1_urlos_RF2 = eye(NRFt_RF2-1);
        C1_urlos_RF2     =     I1_urlos_RF2     +
((gamma_urlos_RF2^2/variance_sistem)*vrfj_urlos_
RF2'*F1_urlos_RF2*vrfj_urlos_RF2); % step 3
        G1_urlos_RF2                               =
((gamma_urlos_RF2^2/variance_sistem)*F1_urlos_RF
2)                                                 -
((gamma_urlos_RF2^4/variance_sistem^2)*F1_urlos_
RF2*vrfj_urlos_RF2*(C1_urlos_RF2^(-
```

```
1))*vrfj_urlos_RF2'*F1_urlos_RF2); % step 4
        sigma_urlos_RF2 = 0;
        for i1_urlos_RF2 = 1:1:num_m % step 5
            for L1_urlos_RF2 = 1:1:num_m
            if L1_urlos_RF2 ~= i1_urlos_RF2
                sigma_urlos_RF2 = sigma_urlos_RF2
+    G1_urlos_RF2(i1_urlos_RF2,L1_urlos_RF2)    *
vrf_urlos_RF2(L1_urlos_RF2,j1_urlos_RF2); % step
6
            end
            end
            nij_urlos_RF2=sigma_urlos_RF2;
            if nij_urlos_RF2 == 0 % step 7

vrf_urlos_RF2(i1_urlos_RF2,j1_urlos_RF2) = 1;
            else

vrf_urlos_RF2(i1_urlos_RF2,j1_urlos_RF2)        =
nij_urlos_RF2/abs(nij_urlos_RF2);
            end
        end
        cekv_RF2 = abs(vrf_urlos_RF2);
    end
    I2_urlos_RF2 = eye(NRFr_RF2);
    VdVdh_urlos_RF2                             =
(sqrt(gamma_urlos_RF2^2))*I2_urlos_RF2;
    Heff_urlos_RF2 = H_urlos*vrf_urlos_RF2;
    Q_urlos_RF2 = vrf_urlos_RF2'*vrf_urlos_RF2;
    svd_urlos_RF2                               =
Heff_urlos_RF2*(Q_urlos_RF2^(-0.5));
    [U_urlos_RF2,S_urlos_RF2,V_urlos_RF2]       =
svd(svd_urlos_RF2);
    Ue_urlos_RF2                                =
V_urlos_RF2(1:Ns_RF2,1:Ns_RF2);
    Te_urlos_RF2                                =
sqrt(P_sistem/NRFt_RF2)*(eye(Ns_RF2));
    Vd_urlos_RF2         =         (Q_urlos_RF2^(-
0.5))*Ue_urlos_RF2*Te_urlos_RF2;

    vrf_old39 = vrf_new39;
```

```
    vrf_new39                                          =
trace(vrf_urlos_RF2*Vd_urlos_RF2*Vd_urlos_RF2'*v
rf_urlos_RF2');
    tol39 = vrf_new39 - vrf_old39;

    jt39 = jt39 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_urlos_RF2 = vrf_urlos_RF2*Vd_urlos_RF2;
F2_urlos_RF2                                       =
H_urlos*Vt_urlos_RF2*Vt_urlos_RF2'*H_urlos';

wrf_urlos_RF2   = ones(num_u,NRFr_RF2); %step 1
l2_urlos_RF2    = length(num_u);
nij2_urlos_RF2  = zeros(num_u,NRFr_RF2);
Wrf_urlos1_RF2  = zeros(num_u,NRFr_RF2);
vrf_new40        = 100;
tol40            = 1;
jt40             = 0;

while (tol40 > 1e-5)
    for j2_urlos_RF2 = 1:1:NRFr_RF2 %step 2
        wrfj_urlos_RF2 = wrf_urlos_RF2;
        wrfj_urlos_RF2(:,j2_urlos_RF2) = [];
        I2_urlos_RF2 = eye(NRFr_RF2-1); %step 3&4
        C2_urlos_RF2     =     I2_urlos_RF2     +
(((1/num_u)/variance_sistem)*wrfj_urlos_RF2'*F2_
urlos_RF2*wrfj_urlos_RF2);
        G2_urlos_RF2                               =
(((1/num_u)/variance_sistem)*F2_urlos_RF2)      -
(((1/num_u)^2/variance_sistem^2)*F2_urlos_RF2*wr
fj_urlos_RF2*(C2_urlos_RF2^(-
1))*wrfj_urlos_RF2'*F2_urlos_RF2);
        sigma2_urlos_RF2 = 0;
        for i2_urlos_RF2 = 1:1:num_u
            for L2_urlos_RF2 = 1:1:num_u
                if L2_urlos_RF2 ~= i2_urlos_RF2
```

```
                    sigma2_urlos_RF2                =
sigma2_urlos_RF2                                    +
G2_urlos_RF2(i2_urlos_RF2,L2_urlos_RF2)             *
wrf_urlos_RF2(L2_urlos_RF2,j2_urlos_RF2);
                end
            end
            nij2_urlos_RF2=sigma2_urlos_RF2;
            if nij2_urlos_RF2 == 0

wrf_urlos_RF2(i2_urlos_RF2,j2_urlos_RF2) = 1;
            else

wrf_urlos_RF2(i2_urlos_RF2,j2_urlos_RF2)            =
nij2_urlos_RF2/abs(nij2_urlos_RF2);
            end
            %i2_urlos_RF2 = i2_urlos + 1;
        end
        %j2_urlos_RF2 = j2_urlos + 1;
    end
    j_URLOS_RF2                                     =
wrf_urlos_RF2'*H_urlos*vrf_urlos_RF2*Vd_urlos_RF
2*Vd_urlos_RF2'*vrf_urlos_RF2'*H_urlos'*wrf_urlo
s_RF2                                               +
variance_sistem*wrf_urlos_RF2'*wrf_urlos_RF2;
    J_URLOS_RF2 = inv(j_URLOS_RF2);
    Wd_urlos_RF2                                    =
J_URLOS_RF2*wrf_urlos_RF2'*H_urlos*vrf_urlos_RF2
*Vd_urlos_RF2;
    Wt_urlos_RF2 = wrf_urlos_RF2*Wd_urlos_RF2;

    vrf_old40 = vrf_new40;
    vrf_new40                                       =
trace(wrf_urlos_RF2*Wd_urlos_RF2*Wd_urlos_RF2'*w
rf_urlos_RF2');
    tol40 = vrf_new40 - vrf_old40;
    jt40 = jt40 + 1;
end

SE_URLOS_RF2(SNRLoop,n_sistem)                      =
real(log2(det((eye(num_u))                          +
```

```
(1/variance_sistem)*Wt_urlos_RF2*(inv(Wt_urlos_R
F2'*Wt_urlos_RF2))*Wt_urlos_RF2'*H_urlos*Vt_urlo
s_RF2*Vt_urlos_RF2'*H_urlos')));

%%%%  MEMBUAT  SISTEM  FULL-DIGITAL  BEAMFORMING
UNTUK USER 1
H_FDB_irf_us1          = H_irf_us2(1:num_u,:);
F1_irf_FDB_US1                                  =
H_FDB_irf_us1'*H_FDB_irf_us1;
gamma_irf_FDB_US1                               =
sqrt(P_sistem/(num_m*NRFt_FDB_US1));
vrf_irf_FDB_US1                                 =
ones(num_m,NRFt_FDB_US1);                % step 1
l1_irf_FDB_US1          = length(num_m);
nij_irf_FDB_US1       = zeros(num_m,NRFt_FDB_US1);
vrf_irf1_FDB_US1      = zeros(num_m,NRFt_FDB_US1);
vrf_new41            = 100;
tol41                = 1;
jt41                 = 0;

while (tol41 > 1e-5)
    for              j1_irf_FDB_US1             =
1:1:NRFt_FDB_US1                           % step
2
        vrfj_irf_FDB_US1                        =
vrf_irf_FDB_US1;
        vrfj_irf_FDB_US1(:,j1_irf_FDB_US1)      =
[];
        I1_irf_FDB_US1                          =
eye(NRFt_FDB_US1-1);
        C1_irf_FDB_US1                          =
I1_irf_FDB_US1                                  +
((gamma_irf_FDB_US1^2/variance_sistem)*vrfj_irf_
FDB_US1'*F1_irf_FDB_US1*vrfj_irf_FDB_US1);
% step 3
        G1_irf_FDB_US1                          =
((gamma_irf_FDB_US1^2/variance_sistem)*F1_irf_FD
B_US1)                                          -
((gamma_irf_FDB_US1^4/variance_sistem^2)*F1_irf_
FDB_US1*vrfj_irf_FDB_US1*(C1_irf_FDB_US1^(-
```

```
1))*vrfj_irf_FDB_US1'*F1_irf_FDB_US1);   % step 4
        sigma_irf_FDB_US1 = 0;
        for           i1_irf_FDB_US1           =
1:1:num_m                        % step 5
            for L1_irf_FDB_US1 = 1:1:num_m
                if       L1_irf_FDB_US1       ~=
i1_irf_FDB_US1
                sigma_irf_FDB_US1               =
sigma_irf_FDB_US1                               +
G1_irf_FDB_US1(i1_irf_FDB_US1,L1_irf_FDB_US1)   *
vrf_irf_FDB_US1(L1_irf_FDB_US1,j1_irf_FDB_US1);
       % step 6
                end
            end
            nij_irf_FDB_US1=sigma_irf_FDB_US1;
            if         nij_irf_FDB_US1         ==
0                        % step 7

vrf_irf_FDB_US1(i1_irf_FDB_US1,j1_irf_FDB_US1)  =
1;
            else

vrf_irf_FDB_US1(i1_irf_FDB_US1,j1_irf_FDB_US1)  =
nij_irf_FDB_US1/abs(nij_irf_FDB_US1);
            end
        end
          cekv_FDB_US1 = abs(vrf_irf_FDB_US1);
    end
    I2_irf_FDB_US1                              =
eye(NRFr_FDB_US1);
    VdVdH_FDB_us1_FDB_US1                       =
(sqrt(gamma_irf_FDB_US1^2))*I2_irf_FDB_US1;
    Heff_irf_FDB_US1                            =
H_FDB_irf_us1*vrf_irf_FDB_US1;
    Q_irf_FDB_US1                               =
vrf_irf_FDB_US1'*vrf_irf_FDB_US1;
    svd_irf_FDB_US1                             =
Heff_irf_FDB_US1*(Q_irf_FDB_US1^(-0.5));
    [U_irf_FDB_US1,S_irf_FDB_US1,V_irf_FDB_US1]
= svd(svd_irf_FDB_US1);
```

```
    Ue_irf_FDB_US1                                  =
V_irf_FDB_US1(1:NRFt_FDB_US1,1:Ns_FDB_US1);
    Te_irf_FDB_US1                                  =
sqrt(P_sistem/NRFt_FDB_US1)*(eye(Ns_FDB_US1));
    Vd_irf_FDB_US1                                  =
(Q_irf_FDB_US1^(-
0.5))*Ue_irf_FDB_US1*Te_irf_FDB_US1;

    vrf_old41 = vrf_new41;
    vrf_new41                                       =
trace(vrf_irf_FDB_US1*Vd_irf_FDB_US1*Vd_irf_FDB_
US1'*vrf_irf_FDB_US1');
    tol41 = vrf_new41 - vrf_old41;

    jt41 = jt41 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_FDB_US1                                      =
vrf_irf_FDB_US1*Vd_irf_FDB_US1;
F2_irf_FDB_US1                                      =
H_FDB_irf_us1*Vt_irf_FDB_US1*Vt_irf_FDB_US1'*H_F
DB_irf_us1';

%P = SNR;
wrf_irf_FDB_US1                                     =
ones(num_u,NRFr_FDB_US1);          %step 1
l2_irf_FDB_US1         = length(num_u);
nij2_irf_FDB_US1     = zeros(num_u,NRFr_FDB_US1);
Wrf_irf1_FDB_US1     = zeros(num_u,NRFr_FDB_US1);
vrf_new42 = 100;
tol42 = 1;
jt42 = 0;

while (tol42 > 1e-5)
    for              j2_irf_FDB_US1                 =
1:1:NRFr_FDB_US1
%step 2
```

```
        wrfj_irf_FDB_US1                                =
wrf_irf_FDB_US1;
        wrfj_irf_FDB_US1(:,j2_irf_FDB_US1)              =
[];
        I2_irf_FDB_US1                                  =
eye(NRFr_FDB_US1-1);    %step 3&4
        C2_irf_FDB_US1                                  =
I2_irf_FDB_US1                                          +
(((1/num_u)/variance_sistem)*wrfj_irf_FDB_US1'*F
2_irf_FDB_US1*wrfj_irf_FDB_US1);
        G2_irf_FDB_US1                                  =
(((1/num_u)/variance_sistem)*F2_irf_FDB_US1)    -
(((1/num_u)^2/variance_sistem^2)*F2_irf_FDB_US1*
wrfj_irf_FDB_US1*(C2_irf_FDB_US1^(-
1))*wrfj_irf_FDB_US1'*F2_irf_FDB_US1);
        sigma2_irf_FDB_US1 = 0;
        for i2_irf_FDB_US1 = 1:1:num_u
            for L2_irf_FDB_US1 = 1:1:num_u
                if       L2_irf_FDB_US1       ~=
i2_irf_FDB_US1
                sigma2_irf_FDB_US1              =
sigma2_irf_FDB_US1                              +
G2_irf_FDB_US1(i2_irf_FDB_US1,L2_irf_FDB_US1)   *
wrf_irf_FDB_US1(L2_irf_FDB_US1,j2_irf_FDB_US1);
                end
            end
            nij2_irf_FDB_US1=sigma2_irf_FDB_US1;
            if nij2_irf_FDB_US1 == 0

wrf_irf_FDB_US1(i2_irf_FDB_US1,j2_irf_FDB_US1)  =
1;
            else

wrf_irf_FDB_US1(i2_irf_FDB_US1,j2_irf_FDB_US1)  =
nij2_irf_FDB_US1/abs(nij2_irf_FDB_US1);
            end
            %i2_irf_FDB_US1 = i2_irf + 1;
        end
      %j2_irf_FDB_US1 = j2_irf + 1;
    end
```

```
    j_IRF_FDB_US1                                   =
wrf_irf_FDB_US1'*H_FDB_irf_us1*vrf_irf_FDB_US1*V
d_irf_FDB_US1*Vd_irf_FDB_US1'*vrf_irf_FDB_US1'*H
_FDB_irf_us1'*wrf_irf_FDB_US1                       +
variance_sistem*wrf_irf_FDB_US1'*wrf_irf_FDB_US1
;
    J_IRF_FDB_US1           = inv(j_IRF_FDB_US1);
    Wd_irf_FDB_US1                                  =
J_IRF_FDB_US1*wrf_irf_FDB_US1'*H_FDB_irf_us1*vrf
_irf_FDB_US1*Vd_irf_FDB_US1;
    Wt_irf_FDB_US1                                  =
wrf_irf_FDB_US1*Wd_irf_FDB_US1;

    vrf_old42 = vrf_new42;
    vrf_new42                                       =
trace(wrf_irf_FDB_US1*Wd_irf_FDB_US1*Wd_irf_FDB_
US1'*wrf_irf_FDB_US1');
    tol42 = vrf_new42 - vrf_old42;

jt42 = jt42 + 1;
end

SE_IRF_FDB_US1(SNRLoop,n_sistem)                    =
real(log2(det((eye(num_u))                          +
(1/variance_sistem)*Wt_irf_FDB_US1*(inv(Wt_irf_F
DB_US1'*Wt_irf_FDB_US1))*Wt_irf_FDB_US1'*H_FDB_i
rf_us1*Vt_irf_FDB_US1*Vt_irf_FDB_US1'*H_FDB_irf_
us1')));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% MEMBUAT SISTEM FULL DIGITAL UNTUK USER 2

H_FDB_irf_us2                                       =
H_irf_us2((num_u+1):total_Rx_US2,:);
F1_irf_FDB_US2                                      =
H_FDB_irf_us2'*H_FDB_irf_us2;
gamma_irf_FDB_US2                                   =
sqrt(P_sistem/(num_m*NRFt_FDB_US2));
vrf_irf_FDB_US2                                     =
```

```
ones(num_m,NRFt_FDB_US2);                % step 1
l1_irf_FDB_US2          = length(num_m);
nij_irf_FDB_US2       = zeros(num_m,NRFt_FDB_US2);
vrf_irf1_FDB_US2      = zeros(num_m,NRFt_FDB_US2);
vrf_new43           = 100;
tol43               = 1;
jt43                = 0;

while (tol43 > 1e-5)
    for              j1_irf_FDB_US2              =
1:1:NRFt_FDB_US2                              % step
2
        vrfj_irf_FDB_US2                        =
vrf_irf_FDB_US2;
        vrfj_irf_FDB_US2(:,j1_irf_FDB_US2)      =
[];
        I1_irf_FDB_US2                          =
eye(NRFt_FDB_US2-1);
        C1_irf_FDB_US2                          =
I1_irf_FDB_US2                                  +
((gamma_irf_FDB_US2^2/variance_sistem)*vrfj_irf_
FDB_US2'*F1_irf_FDB_US2*vrfj_irf_FDB_US2);
% step 3
        G1_irf_FDB_US2                          =
((gamma_irf_FDB_US2^2/variance_sistem)*F1_irf_FD
B_US2)                                          -
((gamma_irf_FDB_US2^4/variance_sistem^2)*F1_irf_
FDB_US2*vrfj_irf_FDB_US2*(C1_irf_FDB_US2^(-
1))*vrfj_irf_FDB_US2'*F1_irf_FDB_US2);   % step 4
        sigma_irf_FDB_US2 = 0;
        for            i1_irf_FDB_US2           =
1:1:num_m                        % step 5
            for L1_irf_FDB_US2 = 1:1:num_m
                if       L1_irf_FDB_US2       ~=
i1_irf_FDB_US2
                sigma_irf_FDB_US2               =
sigma_irf_FDB_US2                               +
G1_irf_FDB_US2(i1_irf_FDB_US2,L1_irf_FDB_US2)   *
vrf_irf_FDB_US2(L1_irf_FDB_US2,j1_irf_FDB_US2);
       % step 6
```

```
                end
            end
            nij_irf_FDB_US2=sigma_irf_FDB_US2;
            if          nij_irf_FDB_US2          ==
0                         % step 7

vrf_irf_FDB_US2(i1_irf_FDB_US2,j1_irf_FDB_US2)  =
1;
            else

vrf_irf_FDB_US2(i1_irf_FDB_US2,j1_irf_FDB_US2)  =
nij_irf_FDB_US2/abs(nij_irf_FDB_US2);
            end
        end
          cekv_FDB_US1 = abs(vrf_irf_FDB_US2);
    end
    I2_irf_FDB_US2                               =
eye(NRFr_FDB_US2);
    VdVdH_FDB_us2_FDB_US2                        =
(sqrt(gamma_irf_FDB_US2^2))*I2_irf_FDB_US2;
    Heff_irf_FDB_US2                             =
H_FDB_irf_us2*vrf_irf_FDB_US2;
    Q_irf_FDB_US2                                =
vrf_irf_FDB_US2'*vrf_irf_FDB_US2;
    svd_irf_FDB_US2                              =
Heff_irf_FDB_US2*(Q_irf_FDB_US2^(-0.5));
    [U_irf_FDB_US2,S_irf_FDB_US2,V_irf_FDB_US2]
= svd(svd_irf_FDB_US2);
    Ue_irf_FDB_US2                               =
V_irf_FDB_US2(1:NRFt_FDB_US2,1:Ns_FDB_US2);
    Te_irf_FDB_US2                               =
sqrt(P_sistem/NRFt_FDB_US2)*(eye(Ns_FDB_US2));
    Vd_irf_FDB_US2                               =
(Q_irf_FDB_US2^(-
0.5))*Ue_irf_FDB_US2*Te_irf_FDB_US2;

    vrf_old43 = vrf_new43;
    vrf_new43                                    =
trace(vrf_irf_FDB_US2*Vd_irf_FDB_US2*Vd_irf_FDB_
US2'*vrf_irf_FDB_US2');
```

```
    tol43 = vrf_new43 - vrf_old43;

    jt43 = jt43 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_FDB_US2                                  =
vrf_irf_FDB_US2*Vd_irf_FDB_US2;
F2_irf_FDB_US2                                  =
H_FDB_irf_us2*Vt_irf_FDB_US2*Vt_irf_FDB_US2'*H_F
DB_irf_us2';

%P = SNR;
wrf_irf_FDB_US2                                 =
ones(num_u,NRFr_FDB_US2);           %step 1
l2_irf_FDB_US2        = length(num_u);
nij2_irf_FDB_US2     = zeros(num_u,NRFr_FDB_US2);
Wrf_irf1_FDB_US2     = zeros(num_u,NRFr_FDB_US2);
vrf_new44 = 100;
tol44 = 1;
jt44 = 0;

while (tol44 > 1e-5)
    for             j2_irf_FDB_US2              =
1:1:NRFr_FDB_US2
%step 2
        wrfj_irf_FDB_US2                        =
wrf_irf_FDB_US2;
        wrfj_irf_FDB_US2(:,j2_irf_FDB_US2)      =
[];
        I2_irf_FDB_US2                          =
eye(NRFr_FDB_US2-1);   %step 3&4
        C2_irf_FDB_US2                          =
I2_irf_FDB_US2                                  +
(((1/num_u)/variance_sistem)*wrfj_irf_FDB_US2'*F
2_irf_FDB_US2*wrfj_irf_FDB_US2);
        G2_irf_FDB_US2                          =
(((1/num_u)/variance_sistem)*F2_irf_FDB_US2)    -
```

```
(((1/num_u)^2/variance_sistem^2)*F2_irf_FDB_US2*
wrfj_irf_FDB_US2*(C2_irf_FDB_US2^(-
1))*wrfj_irf_FDB_US2'*F2_irf_FDB_US2);
        sigma2_IRF_FDB_US2 = 0;
        for i2_irf_FDB_US2 = 1:1:num_u
            for L2_irf_FDB_US2 = 1:1:num_u
                if       L2_irf_FDB_US2       ~=
i2_irf_FDB_US2
                sigma2_IRF_FDB_US2             =
sigma2_IRF_FDB_US2                             +
G2_irf_FDB_US2(i2_irf_FDB_US2,L2_irf_FDB_US2)  *
wrf_irf_FDB_US2(L2_irf_FDB_US2,j2_irf_FDB_US2);
                end
            end
            nij2_irf_FDB_US2=sigma2_IRF_FDB_US2;
            if nij2_irf_FDB_US2 == 0

wrf_irf_FDB_US2(i2_irf_FDB_US2,j2_irf_FDB_US2) =
1;
            else

wrf_irf_FDB_US2(i2_irf_FDB_US2,j2_irf_FDB_US2) =
nij2_irf_FDB_US2/abs(nij2_irf_FDB_US2);
            end
            %i2_irf_FDB_US2 = i2_irf + 1;
        end
      %j2_irf_FDB_US2 = j2_irf + 1;
    end
    j_IRF_FDB_US2                              =
wrf_irf_FDB_US2'*H_FDB_irf_us2*vrf_irf_FDB_US2*V
d_irf_FDB_US2*Vd_irf_FDB_US2'*vrf_irf_FDB_US2'*H
_FDB_irf_us2'*wrf_irf_FDB_US2                  +
variance_sistem*wrf_irf_FDB_US2'*wrf_irf_FDB_US2
;
    J_IRF_FDB_US2          = inv(j_IRF_FDB_US2);
    Wd_irf_FDB_US2                             =
J_IRF_FDB_US2*wrf_irf_FDB_US2'*H_FDB_irf_us2*vrf
_irf_FDB_US2*Vd_irf_FDB_US2;
    Wt_irf_FDB_US2                             =
wrf_irf_FDB_US2*Wd_irf_FDB_US2;
```

```
    vrf_old44 = vrf_new44;
    vrf_new44                                          =
trace(wrf_irf_FDB_US2*Wd_irf_FDB_US2*Wd_irf_FDB_
US2'*wrf_irf_FDB_US2');
    tol44 = vrf_new44 - vrf_old44;

jt44 = jt44 + 1;
end

SE_IRF_FDB_US2(SNRLoop,n_sistem)                       =
real(log2(det((eye(num_u))                             +
(1/variance_sistem)*Wt_irf_FDB_US2*(inv(Wt_irf_F
DB_US2'*Wt_irf_FDB_US2))*Wt_irf_FDB_US2'*H_FDB_i
rf_us2*Vt_irf_FDB_US2*Vt_irf_FDB_US2'*H_FDB_irf_
us2')));

SE_IRF_FDB2us(SNRLoop,n_sistem)                        =
(SE_IRF_FDB_US1(SNRLoop,n_sistem)                      +
SE_IRF_FDB_US2(SNRLoop,n_sistem))/2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% MEMBUAT SISTEM DENGAN HYBRID BEAMFORMING
UNTUK USER 1

H_HB_irf_us1    = H_irf_us2(1:num_u,:);
F1_irf_HB_US1   = H_HB_irf_us1'*H_HB_irf_us1;
gamma_irf_HB_US1                                       =
sqrt(P_sistem/(num_m*NRFt_HB_US1));
vrf_irf_HB_US1  = ones(num_m,NRFt_HB_US1); % step
1
l1_irf_HB_US1   = length(num_m);
nij_irf_HB_US1  = zeros(num_m,NRFt_HB_US1);
vrf_irf1_HB_US1 = zeros(num_m,NRFt_HB_US1);
vrf_new45 = 100;
tol45 = 1;
jt45 = 0;

while (tol45 > 1e-5)
```

```
    for j1_irf_HB_US1 = 1:1:NRFt_HB_US1 % step 2
        vrfj_irf_HB_US1 = vrf_irf_HB_US1;
        vrfj_irf_HB_US1(:,j1_irf_HB_US1) = [];
        I1_irf_HB_US1 = eye(NRFt_HB_US1-1);
        C1_irf_HB_US1     =     I1_irf_HB_US1     +
((gamma_irf_HB_US1^2/variance_sistem)*vrfj_irf_H
B_US1'*F1_irf_HB_US1*vrfj_irf_HB_US1); % step 3
        G1_irf_HB_US1                             =
((gamma_irf_HB_US1^2/variance_sistem)*F1_irf_HB_
US1)                                              -
((gamma_irf_HB_US1^4/variance_sistem^2)*F1_irf_H
B_US1*vrfj_irf_HB_US1*(C1_irf_HB_US1^(-
1))*vrfj_irf_HB_US1'*F1_irf_HB_US1); % step 4
        sigma_irf_HB_US1 = 0;
        for i1_irf_HB_US1 = 1:1:num_m % step 5
            for L1_irf_HB_US1 = 1:1:num_m
                if L1_irf_HB_US1 ~= i1_irf_HB_US1
                    sigma_irf_HB_US1              =
sigma_irf_HB_US1                                  +
G1_irf_HB_US1(i1_irf_HB_US1,L1_irf_HB_US1)        *
vrf_irf_HB_US1(L1_irf_HB_US1,j1_irf_HB_US1);      %
step 6
                end
            end
            nij_irf_HB_US1=sigma_irf_HB_US1;
            if nij_irf_HB_US1 == 0 % step 7

vrf_irf_HB_US1(i1_irf_HB_US1,j1_irf_HB_US1) = 1;
            else

vrf_irf_HB_US1(i1_irf_HB_US1,j1_irf_HB_US1)       =
nij_irf_HB_US1/abs(nij_irf_HB_US1);
            end
        end
        cekv_HB_US2 = abs(vrf_irf_HB_US1);
    end
    I2_irf_HB_US1 = eye(NRFr_HB_US1);
    VdVdH_HB_us1_HB_US1                           =
(sqrt(gamma_irf_HB_US1^2))*I2_irf_HB_US1;
    Heff_irf_HB_US1                               =
```

```
H_HB_irf_us1*vrf_irf_HB_US1;
    Q_irf_HB_US1                                    =
vrf_irf_HB_US1'*vrf_irf_HB_US1;
    svd_irf_HB_US1                                  =
Heff_irf_HB_US1*(Q_irf_HB_US1^(-0.5));
    [U_irf_HB_US1,S_irf_HB_US1,V_irf_HB_US1]    =
svd(svd_irf_HB_US1);
    Ue_irf_HB_US1                                   =
V_irf_HB_US1(1:Ns_HB_US1,1:Ns_HB_US1);
    Te_irf_HB_US1                                   =
sqrt(P_sistem/NRFt_HB_US1)*(eye(Ns_HB_US1));
    Vd_irf_HB_US1       =       (Q_irf_HB_US1^(-
0.5))*Ue_irf_HB_US1*Te_irf_HB_US1;

    vrf_old45 = vrf_new45;
    vrf_new45                                       =
trace(vrf_irf_HB_US1*Vd_irf_HB_US1*Vd_irf_HB_US1
'*vrf_irf_HB_US1');
    tol45 = vrf_new45 - vrf_old45;
    jt45 = jt45 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO
Vt_irf_HB_US1 = vrf_irf_HB_US1*Vd_irf_HB_US1;
F2_irf_HB_US1                                       =
H_HB_irf_us1*Vt_irf_HB_US1*Vt_irf_HB_US1'*H_HB_i
rf_us1';

wrf_irf_HB_US1 = ones(num_u,NRFr_HB_US1); %step 1
l2_irf_HB_US1 = length(num_u);
nij2_irf_HB_US1 = zeros(num_u,NRFr_HB_US1);
Wrf_irf1_HB_US1 = zeros(num_u,NRFr_HB_US1);
vrf_new46 = 100;
tol46 = 1;
jt46 = 0;

while (tol46 > 1e-5)
    for j2_irf_HB_US1 = 1:1:NRFr_HB_US1 %step 2
        wrfj_irf_HB_US1 = wrf_irf_HB_US1;
```

```
        wrfj_irf_HB_US1(:,j2_irf_HB_US1) = [];
        I2_irf_HB_US1 = eye(NRFr_HB_US1-1); %step
3&4
        C2_irf_HB_US1    =    I2_irf_HB_US1    +
(((1/num_u)/variance_sistem)*wrfj_irf_HB_US1'*F2
_irf_HB_US1*wrfj_irf_HB_US1);
        G2_irf_HB_US1                                =
(((1/num_u)/variance_sistem)*F2_irf_HB_US1)     -
(((1/num_u)^2/variance_sistem^2)*F2_irf_HB_US1*w
rfj_irf_HB_US1*(C2_irf_HB_US1^(-
1))*wrfj_irf_HB_US1'*F2_irf_HB_US1);
        sigma2_irf_HB_US1 = 0;
        for i2_irf_HB_US1 = 1:1:num_u
            for L2_irf_HB_US1 = 1:1:num_u
                if L2_irf_HB_US1 ~= i2_irf_HB_US1
                    sigma2_irf_HB_US1           =
sigma2_irf_HB_US1                               +
G2_irf_HB_US1(i2_irf_HB_US1,L2_irf_HB_US1)      *
wrf_irf_HB_US1(L2_irf_HB_US1,j2_irf_HB_US1);
                end
            end
            nij2_irf_HB_US1=sigma2_irf_HB_US1;
            if nij2_irf_HB_US1 == 0

wrf_irf_HB_US1(i2_irf_HB_US1,j2_irf_HB_US1) = 1;
            else

wrf_irf_HB_US1(i2_irf_HB_US1,j2_irf_HB_US1)     =
nij2_irf_HB_US1/abs(nij2_irf_HB_US1);
            end
            %i2_irf_HB_US1 = i2_irf + 1;
        end
        %j2_irf_HB_US1 = j2_irf + 1;
    end
    j_IRF_HB_US1                                 =
wrf_irf_HB_US1'*H_HB_irf_us1*vrf_irf_HB_US1*Vd_i
rf_HB_US1*Vd_irf_HB_US1'*vrf_irf_HB_US1'*H_HB_ir
f_us1'*wrf_irf_HB_US1                            +
variance_sistem*wrf_irf_HB_US1'*wrf_irf_HB_US1;
    J_IRF_HB_US1 = inv(j_IRF_HB_US1);
```

```
    Wd_irf_HB_US1                                   =
J_IRF_HB_US1*wrf_irf_HB_US1'*H_HB_irf_us1*vrf_ir
f_HB_US1*Vd_irf_HB_US1;
    Wt_irf_HB_US1 = wrf_irf_HB_US1*Wd_irf_HB_US1;

    vrf_old46 = vrf_new46;
    vrf_new46                                       =
trace(wrf_irf_HB_US1*Wd_irf_HB_US1*Wd_irf_HB_US1
'*wrf_irf_HB_US1');
    tol46 = vrf_new46 - vrf_old46;

    jt46 = jt46 + 1;
end

SE_IRF_HB_US1(SNRLoop,n_sistem)                     =
real(log2(det((eye(num_u))                          +
(1/variance_sistem)*Wt_irf_HB_US1*(inv(Wt_irf_HB
_US1'*Wt_irf_HB_US1))*Wt_irf_HB_US1'*H_HB_irf_us
1*Vt_irf_HB_US1*Vt_irf_HB_US1'*H_HB_irf_us1')));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% MEMBUAT SISTEM HYBRID BEAMFORMING UNTUK USER
2

H_HB_irf_us2                                        =
H_irf_us2((num_u+1):total_Rx_US2,:);
F1_irf_HB_US2    = H_HB_irf_us2'*H_HB_irf_us2;
gamma_irf_HB_US2                                    =
sqrt(P_sistem/(num_m*NRFt_HB_US2));
vrf_irf_HB_US2   = ones(num_m,NRFt_HB_US2); % step
1
l1_irf_HB_US2    = length(num_m);
nij_irf_HB_US2   = zeros(num_m,NRFt_HB_US2);
vrf_irf1_HB_US2 = zeros(num_m,NRFt_HB_US2);
vrf_new47 = 100;
tol47 = 1;
jt47 = 0;

while (tol47 > 1e-5)
```

```
    for j1_irf_HB_US2 = 1:1:NRFt_HB_US2 % step 2
        vrfj_irf_HB_US2 = vrf_irf_HB_US2;
        vrfj_irf_HB_US2(:,j1_irf_HB_US2) = [];
        I1_irf_HB_US2 = eye(NRFt_HB_US2-1);
        C1_irf_HB_US2     =     I1_irf_HB_US2     +
((gamma_irf_HB_US2^2/variance_sistem)*vrfj_irf_H
B_US2'*F1_irf_HB_US2*vrfj_irf_HB_US2); % step 3
        G1_irf_HB_US2                             =
((gamma_irf_HB_US2^2/variance_sistem)*F1_irf_HB_
US2)                                              -
((gamma_irf_HB_US2^4/variance_sistem^2)*F1_irf_H
B_US2*vrfj_irf_HB_US2*(C1_irf_HB_US2^(-
1))*vrfj_irf_HB_US2'*F1_irf_HB_US2); % step 4
        sigma_irf_HB_US2 = 0;
        for i1_irf_HB_US2 = 1:1:num_m % step 5
            for L1_irf_HB_US2 = 1:1:num_m
                if L1_irf_HB_US2 ~= i1_irf_HB_US2
                    sigma_irf_HB_US2              =
sigma_irf_HB_US2                                  +
G1_irf_HB_US2(i1_irf_HB_US2,L1_irf_HB_US2)        *
vrf_irf_HB_US2(L1_irf_HB_US2,j1_irf_HB_US2);      %
step 6
                end
            end
            nij_irf_HB_US2=sigma_irf_HB_US2;
            if nij_irf_HB_US2 == 0 % step 7

vrf_irf_HB_US2(i1_irf_HB_US2,j1_irf_HB_US2) = 1;
            else

vrf_irf_HB_US2(i1_irf_HB_US2,j1_irf_HB_US2)       =
nij_irf_HB_US2/abs(nij_irf_HB_US2);
            end
        end
        cekv_HB_US2 = abs(vrf_irf_HB_US2);
    end
    I2_irf_HB_US2 = eye(NRFr_HB_US2);
    VdVdH_HB_us2_HB_US2                           =
(sqrt(gamma_irf_HB_US2^2))*I2_irf_HB_US2;
    Heff_irf_HB_US2                               =
```

```
H_HB_irf_us2*vrf_irf_HB_US2;
    Q_irf_HB_US2                                    =
vrf_irf_HB_US2'*vrf_irf_HB_US2;
    svd_irf_HB_US2                                  =
Heff_irf_HB_US2*(Q_irf_HB_US2^(-0.5));
    [U_irf_HB_US2,S_irf_HB_US2,V_irf_HB_US2]    =
svd(svd_irf_HB_US2);
    Ue_irf_HB_US2                                   =
V_irf_HB_US2(1:Ns_HB_US2,1:Ns_HB_US2);
    Te_irf_HB_US2                                   =
sqrt(P_sistem/NRFt_HB_US2)*(eye(Ns_HB_US2));
    Vd_irf_HB_US2       =       (Q_irf_HB_US2^(-
0.5))*Ue_irf_HB_US2*Te_irf_HB_US2;

    vrf_old47 = vrf_new47;
    vrf_new47                                       =
trace(vrf_irf_HB_US2*Vd_irf_HB_US2*Vd_irf_HB_US2
'*vrf_irf_HB_US2');
    tol47 = vrf_new47 - vrf_old47;

    jt47 = jt47 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO

Vt_irf_HB_US2 = vrf_irf_HB_US2*Vd_irf_HB_US2;
F2_irf_HB_US2                                       =
H_HB_irf_us2*Vt_irf_HB_US2*Vt_irf_HB_US2'*H_HB_i
rf_us2';

wrf_irf_HB_US2 = ones(num_u,NRFr_HB_US2); %step 1
l2_irf_HB_US2 = length(num_u);
nij2_irf_HB_US2 = zeros(num_u,NRFr_HB_US2);
Wrf_irf1_HB_US2 = zeros(num_u,NRFr_HB_US2);
vrf_new48 = 100;
tol48 = 1;
jt48 = 0;

while (tol48 > 1e-5)
```

```
    for j2_irf_HB_US2 = 1:1:NRFr_HB_US2 %step 2
        wrfj_irf_HB_US2 = wrf_irf_HB_US2;
        wrfj_irf_HB_US2(:,j2_irf_HB_US2) = [];
        I2_irf_HB_US2 = eye(NRFr_HB_US2-1); %step
3&4
        C2_irf_HB_US2    =    I2_irf_HB_US2    +
(((1/num_u)/variance_sistem)*wrfj_irf_HB_US2'*F2
_irf_HB_US2*wrfj_irf_HB_US2);
        G2_irf_HB_US2                                =
(((1/num_u)/variance_sistem)*F2_irf_HB_US2)     -
(((1/num_u)^2/variance_sistem^2)*F2_irf_HB_US2*w
rfj_irf_HB_US2*(C2_irf_HB_US2^(-
1))*wrfj_irf_HB_US2'*F2_irf_HB_US2);
        sigma2_irf_HB_US2 = 0;
        for i2_irf_HB_US2 = 1:1:num_u
            for L2_irf_HB_US2 = 1:1:num_u
                if L2_irf_HB_US2 ~= i2_irf_HB_US2
                    sigma2_irf_HB_US2          =
sigma2_irf_HB_US2                               +
G2_irf_HB_US2(i2_irf_HB_US2,L2_irf_HB_US2)      *
wrf_irf_HB_US2(L2_irf_HB_US2,j2_irf_HB_US2);
                end
            end
            nij2_irf_HB_US2=sigma2_irf_HB_US2;
            if nij2_irf_HB_US2 == 0

wrf_irf_HB_US2(i2_irf_HB_US2,j2_irf_HB_US2) = 1;
            else

wrf_irf_HB_US2(i2_irf_HB_US2,j2_irf_HB_US2)     =
nij2_irf_HB_US2/abs(nij2_irf_HB_US2);
            end
            %i2_irf_HB_US2 = i2_irf + 1;
        end
        %j2_irf_HB_US2 = j2_irf + 1;
    end
    j_IRF_HB_US2                                 =
wrf_irf_HB_US2'*H_HB_irf_us2*vrf_irf_HB_US2*Vd_i
rf_HB_US2*Vd_irf_HB_US2'*vrf_irf_HB_US2'*H_HB_ir
f_us2'*wrf_irf_HB_US2                            +
```

```
variance_sistem*wrf_irf_HB_US2'*wrf_irf_HB_US2;
    J_IRF_HB_US2 = inv(j_IRF_HB_US2);
    Wd_irf_HB_US2                                    =
J_IRF_HB_US2*wrf_irf_HB_US2'*H_HB_irf_us2*vrf_ir
f_HB_US2*Vd_irf_HB_US2;
    Wt_irf_HB_US2 = wrf_irf_HB_US2*Wd_irf_HB_US2;

    vrf_old48 = vrf_new48;
    vrf_new48                                        =
trace(wrf_irf_HB_US2*Wd_irf_HB_US2*Wd_irf_HB_US2
'*wrf_irf_HB_US2');
    tol48 = vrf_new48 - vrf_old48;
    jt48 = jt48 + 1;
end

SE_IRF_HB_US2(SNRLoop,n_sistem)                      =
real(log2(det((eye(num_u))                           +
(1/variance_sistem)*Wt_irf_HB_US2*(inv(Wt_irf_HB
_US2'*Wt_irf_HB_US2))*Wt_irf_HB_US2'*H_HB_irf_us
2*Vt_irf_HB_US2*Vt_irf_HB_US2'*H_HB_irf_us2')));

SE_IRF_HB2us(SNRLoop,n_sistem)                       =
(SE_IRF_HB_US1(SNRLoop,n_sistem)                     +
SE_IRF_HB_US2(SNRLoop,n_sistem))/2;

%%%%  MEMBUAT  SISTEM  FULL-DIGITAL  BEAMFORMING
UNTUK USER 1
H_FDB_urlos_us1                                      =
H_urlos_us2(1:num_u,:);
F1_urlos_FDB_US1                                     =
H_FDB_urlos_us1'*H_FDB_urlos_us1;
gamma_urlos_FDB_US1                                  =
sqrt(P_sistem/(num_m*NRFt_FDB_US1));
vrf_urlos_FDB_US1                                    =
ones(num_m,NRFt_FDB_US1);                  % step 1
l1_urlos_FDB_US1         = length(num_m);
nij_urlos_FDB_US1                                    =
zeros(num_m,NRFt_FDB_US1);
vrf_urlos1_FDB_US1                                   =
zeros(num_m,NRFt_FDB_US1);
```

```
vrf_new49 = 100;
tol49 = 1;
jt49 = 0;

while (tol49 > 1e-5)
    for             j1_urlos_FDB_US1             =
1:1:NRFt_FDB_US1                                % step
2
        vrfj_urlos_FDB_US1                       =
vrf_urlos_FDB_US1;
        vrfj_urlos_FDB_US1(:,j1_urlos_FDB_US1)
= [];
        I1_urlos_FDB_US1                         =
eye(NRFt_FDB_US1-1);
        C1_urlos_FDB_US1                         =
I1_urlos_FDB_US1                                 +
((gamma_urlos_FDB_US1^2/variance_sistem)*vrfj_ur
los_FDB_US1'*F1_urlos_FDB_US1*vrfj_urlos_FDB_US1
);           % step 3
        G1_urlos_FDB_US1                         =
((gamma_urlos_FDB_US1^2/variance_sistem)*F1_urlo
s_FDB_US1)                                       -
((gamma_urlos_FDB_US1^4/variance_sistem^2)*F1_ur
los_FDB_US1*vrfj_urlos_FDB_US1*(C1_urlos_FDB_US1
^(-1))*vrfj_urlos_FDB_US1'*F1_urlos_FDB_US1);   %
step 4
        sigma_urlos_FDB_US1 = 0;
        for          i1_urlos_FDB_US1            =
1:1:num_m                       % step 5
            for L1_urlos_FDB_US1 = 1:1:num_m
                if      L1_urlos_FDB_US1       ~=
i1_urlos_FDB_US1
                sigma_urlos_FDB_US1              =
sigma_urlos_FDB_US1                              +
G1_urlos_FDB_US1(i1_urlos_FDB_US1,L1_urlos_FDB_U
S1)                                              *
vrf_urlos_FDB_US1(L1_urlos_FDB_US1,j1_urlos_FDB_
US1);                  % step 6
                end
            end
```

```
nij_urlos_FDB_US1=sigma_urlos_FDB_US1;
            if        nij_urlos_FDB_US1        ==
0                         % step 7

vrf_urlos_FDB_US1(i1_urlos_FDB_US1,j1_urlos_FDB_
US1) = 1;
            else

vrf_urlos_FDB_US1(i1_urlos_FDB_US1,j1_urlos_FDB_
US1) = nij_urlos_FDB_US1/abs(nij_urlos_FDB_US1);
            end
         end
           cekv_FDB_US1 = abs(vrf_urlos_FDB_US1);
    end
    I2_urlos_FDB_US1                              =
eye(NRFr_FDB_US1);
    VdVdH_FDB_us1_FDB_US1                         =
(sqrt(gamma_urlos_FDB_US1^2))*I2_urlos_FDB_US1;
    Heff_urlos_FDB_US1                            =
H_FDB_urlos_us1*vrf_urlos_FDB_US1;
    Q_urlos_FDB_US1                               =
vrf_urlos_FDB_US1'*vrf_urlos_FDB_US1;
    svd_urlos_FDB_US1                             =
Heff_urlos_FDB_US1*(Q_urlos_FDB_US1^(-0.5));

[U_urlos_FDB_US1,S_urlos_FDB_US1,V_urlos_FDB_US1
]     = svd(svd_urlos_FDB_US1);
    Ue_urlos_FDB_US1                              =
V_urlos_FDB_US1(1:NRFt_FDB_US1,1:Ns_FDB_US1);
    Te_urlos_FDB_US1                              =
sqrt(P_sistem/NRFt_FDB_US1)*(eye(Ns_FDB_US1));
    Vd_urlos_FDB_US1                              =
(Q_urlos_FDB_US1^(-
0.5))*Ue_urlos_FDB_US1*Te_urlos_FDB_US1;

    vrf_old49 = vrf_new49;
    vrf_new49                                     =
trace(vrf_urlos_FDB_US1*Vd_urlos_FDB_US1*Vd_urlo
s_FDB_US1'*vrf_urlos_FDB_US1');
```

```
    tol49 = vrf_new49 - vrf_old49;

    jt49 = jt49 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO
Vt_urlos_FDB_US1                                =
vrf_urlos_FDB_US1*Vd_urlos_FDB_US1;
F2_urlos_FDB_US1                                =
H_FDB_urlos_us1*Vt_urlos_FDB_US1*Vt_urlos_FDB_US
1'*H_FDB_urlos_us1';

wrf_urlos_FDB_US1                               =
ones(num_u,NRFr_FDB_US1);          %step 1
l2_urlos_FDB_US1         = length(num_u);
nij2_urlos_FDB_US1                              =
zeros(num_u,NRFr_FDB_US1);
Wrf_urlos1_FDB_US1                              =
zeros(num_u,NRFr_FDB_US1);
vrf_new50 = 100;
tol50 = 1;
jt50 = 0;

while (tol50 > 1e-5)
    for             j2_urlos_FDB_US1            =
1:1:NRFr_FDB_US1
%step 2
        wrfj_urlos_FDB_US1                      =
wrf_urlos_FDB_US1;
        wrfj_urlos_FDB_US1(:,j2_urlos_FDB_US1)
= [];
        I2_urlos_FDB_US1                        =
eye(NRFr_FDB_US1-1);   %step 3&4
        C2_urlos_FDB_US1                        =
I2_urlos_FDB_US1                                +
(((1/num_u)/variance_sistem)*wrfj_urlos_FDB_US1'
*F2_urlos_FDB_US1*wrfj_urlos_FDB_US1);
        G2_urlos_FDB_US1                        =
(((1/num_u)/variance_sistem)*F2_urlos_FDB_US1)  -
```

```
(((1/num_u)^2/variance_sistem^2)*F2_urlos_FDB_US
1*wrfj_urlos_FDB_US1*(C2_urlos_FDB_US1^(-
1))*wrfj_urlos_FDB_US1'*F2_urlos_FDB_US1);
        sigma2_urlos_FDB_US1 = 0;
        for i2_urlos_FDB_US1 = 1:1:num_u
            for L2_urlos_FDB_US1 = 1:1:num_u
                if      L2_urlos_FDB_US1      ~=
i2_urlos_FDB_US1
                sigma2_urlos_FDB_US1           =
sigma2_urlos_FDB_US1                           +
G2_urlos_FDB_US1(i2_urlos_FDB_US1,L2_urlos_FDB_U
S1)                                            *
wrf_urlos_FDB_US1(L2_urlos_FDB_US1,j2_urlos_FDB_
US1);
                end
            end

nij2_urlos_FDB_US1=sigma2_urlos_FDB_US1;
            if nij2_urlos_FDB_US1 == 0

wrf_urlos_FDB_US1(i2_urlos_FDB_US1,j2_urlos_FDB_
US1) = 1;
            else

wrf_urlos_FDB_US1(i2_urlos_FDB_US1,j2_urlos_FDB_
US1)                                           =
nij2_urlos_FDB_US1/abs(nij2_urlos_FDB_US1);
            end
            %i2_urlos_FDB_US1 = i2_urlos + 1;
        end
      %j2_urlos_FDB_US1 = j2_urlos + 1;
    end
    j_urlos_FDB_US1                            =
wrf_urlos_FDB_US1'*H_FDB_urlos_us1*vrf_urlos_FDB
_US1*Vd_urlos_FDB_US1*Vd_urlos_FDB_US1'*vrf_urlo
s_FDB_US1'*H_FDB_urlos_us1'*wrf_urlos_FDB_US1   +
variance_sistem*wrf_urlos_FDB_US1'*wrf_urlos_FDB
_US1;
    J_urlos_FDB_US1                            =
inv(j_urlos_FDB_US1);
```

```
    Wd_urlos_FDB_US1                                        =
J_urlos_FDB_US1*wrf_urlos_FDB_US1'*H_FDB_urlos_u
s1*vrf_urlos_FDB_US1*Vd_urlos_FDB_US1;
    Wt_urlos_FDB_US1                                        =
wrf_urlos_FDB_US1*Wd_urlos_FDB_US1;

    vrf_old50 = vrf_new50;
    vrf_new50                                               =
trace(wrf_urlos_FDB_US1*Wd_urlos_FDB_US1*Wd_urlo
s_FDB_US1'*wrf_urlos_FDB_US1');
    tol50 = vrf_new50 - vrf_old50;

jt50 = jt50 + 1;
end

SE_urlos_FDB_US1(SNRLoop,n_sistem)                          =
real(log2(det((eye(num_u))                                  +
(1/variance_sistem)*Wt_urlos_FDB_US1*(inv(Wt_url
os_FDB_US1'*Wt_urlos_FDB_US1))*Wt_urlos_FDB_US1'
*H_FDB_urlos_us1*Vt_urlos_FDB_US1*Vt_urlos_FDB_U
S1'*H_FDB_urlos_us1'))));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%  MEMBUAT SISTEM HYBRID BEAMFORMING

H_FDB_urlos_us2                                             =
H_urlos_us2((num_u+1):total_Rx_US2,:);
F1_urlos_FDB_US2                                            =
H_FDB_urlos_us2'*H_FDB_urlos_us2;
gamma_urlos_FDB_US2                                         =
sqrt(P_sistem/(num_m*NRFt_FDB_US2));
vrf_urlos_FDB_US2                                           =
ones(num_m,NRFt_FDB_US2);                       % step 1
l1_urlos_FDB_US2        = length(num_m);
nij_urlos_FDB_US2                                           =
zeros(num_m,NRFt_FDB_US2);
vrf_urlos1_FDB_US2                                          =
zeros(num_m,NRFt_FDB_US2);
vrf_new51         = 100;
```

```
tol51               = 1;
jt51                = 0;

while (tol51 > 1e-5)
    for             j1_urlos_FDB_US2            =
1:1:NRFt_FDB_US2                              % step
2
        vrfj_urlos_FDB_US2                      =
vrf_urlos_FDB_US2;
        vrfj_urlos_FDB_US2(:,j1_urlos_FDB_US2)
= [];
        I1_urlos_FDB_US2                        =
eye(NRFt_FDB_US2-1);
        C1_urlos_FDB_US2                        =
I1_urlos_FDB_US2                                +
((gamma_urlos_FDB_US2^2/variance_sistem)*vrfj_ur
los_FDB_US2'*F1_urlos_FDB_US2*vrfj_urlos_FDB_US2
);            % step 3
        G1_urlos_FDB_US2                        =
((gamma_urlos_FDB_US2^2/variance_sistem)*F1_urlo
s_FDB_US2)                                      -
((gamma_urlos_FDB_US2^4/variance_sistem^2)*F1_ur
los_FDB_US2*vrfj_urlos_FDB_US2*(C1_urlos_FDB_US2
^(-1))*vrfj_urlos_FDB_US2'*F1_urlos_FDB_US2);   %
step 4
        sigma_urlos_FDB_US2 = 0;
        for           i1_urlos_FDB_US2          =
1:1:num_m                       % step 5
            for L1_urlos_FDB_US2 = 1:1:num_m
                if       L1_urlos_FDB_US2       ~=
i1_urlos_FDB_US2
                sigma_urlos_FDB_US2             =
sigma_urlos_FDB_US2                             +
G1_urlos_FDB_US2(i1_urlos_FDB_US2,L1_urlos_FDB_U
S2)                                             *
vrf_urlos_FDB_US2(L1_urlos_FDB_US2,j1_urlos_FDB_
US2);                 % step 6
                end
            end
```

```
nij_urlos_FDB_US2=sigma_urlos_FDB_US2;
            if        nij_urlos_FDB_US2        ==
0                        % step 7

vrf_urlos_FDB_US2(i1_urlos_FDB_US2,j1_urlos_FDB_
US2) = 1;
            else

vrf_urlos_FDB_US2(i1_urlos_FDB_US2,j1_urlos_FDB_
US2) = nij_urlos_FDB_US2/abs(nij_urlos_FDB_US2);
            end
         end
           cekv_FDB_US1 = abs(vrf_urlos_FDB_US2);
    end
    I2_urlos_FDB_US2                              =
eye(NRFr_FDB_US2);
    VdVdH_FDB_us2_FDB_US2                         =
(sqrt(gamma_urlos_FDB_US2^2))*I2_urlos_FDB_US2;
    Heff_urlos_FDB_US2                            =
H_FDB_urlos_us2*vrf_urlos_FDB_US2;
    Q_urlos_FDB_US2                               =
vrf_urlos_FDB_US2'*vrf_urlos_FDB_US2;
    svd_urlos_FDB_US2                             =
Heff_urlos_FDB_US2*(Q_urlos_FDB_US2^(-0.5));

[U_urlos_FDB_US2,S_urlos_FDB_US2,V_urlos_FDB_US2
]     = svd(svd_urlos_FDB_US2);
    Ue_urlos_FDB_US2                              =
V_urlos_FDB_US2(1:NRFt_FDB_US2,1:Ns_FDB_US2);
    Te_urlos_FDB_US2                              =
sqrt(P_sistem/NRFt_FDB_US2)*(eye(Ns_FDB_US2));
    Vd_urlos_FDB_US2                              =
(Q_urlos_FDB_US2^(-
0.5))*Ue_urlos_FDB_US2*Te_urlos_FDB_US2;

    vrf_old51 = vrf_new51;
    vrf_new51                                     =
trace(vrf_urlos_FDB_US2*Vd_urlos_FDB_US2*Vd_urlo
s_FDB_US2'*vrf_urlos_FDB_US2');
    tol51 = vrf_new51 - vrf_old51;
```

```
    jt51 = jt51 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO
Vt_urlos_FDB_US2                                =
vrf_urlos_FDB_US2*Vd_urlos_FDB_US2;
F2_urlos_FDB_US2                                =
H_FDB_urlos_us2*Vt_urlos_FDB_US2*Vt_urlos_FDB_US
2'*H_FDB_urlos_us2';

wrf_urlos_FDB_US2                               =
ones(num_u,NRFr_FDB_US2);          %step 1
l2_urlos_FDB_US2        = length(num_u);
nij2_urlos_FDB_US2                              =
zeros(num_u,NRFr_FDB_US2);
Wrf_urlos1_FDB_US2                              =
zeros(num_u,NRFr_FDB_US2);
vrf_new52 = 100;
tol52 = 1;
jt52 = 0;

while (tol52 > 1e-5)
    for             j2_urlos_FDB_US2            =
1:1:NRFr_FDB_US2
%step 2
        wrfj_urlos_FDB_US2                      =
wrf_urlos_FDB_US2;
        wrfj_urlos_FDB_US2(:,j2_urlos_FDB_US2)
= [];
        I2_urlos_FDB_US2                        =
eye(NRFr_FDB_US2-1);   %step 3&4
        C2_urlos_FDB_US2                        =
I2_urlos_FDB_US2                                +
(((1/num_u)/variance_sistem)*wrfj_urlos_FDB_US2'
*F2_urlos_FDB_US2*wrfj_urlos_FDB_US2);
        G2_urlos_FDB_US2                        =
(((1/num_u)/variance_sistem)*F2_urlos_FDB_US2)  -
(((1/num_u)^2/variance_sistem^2)*F2_urlos_FDB_US
```

```
2*wrfj_urlos_FDB_US2*(C2_urlos_FDB_US2^(-
1))*wrfj_urlos_FDB_US2'*F2_urlos_FDB_US2);
        sigma2_urlos_FDB_US2 = 0;
        for i2_urlos_FDB_US2 = 1:1:num_u
            for L2_urlos_FDB_US2 = 1:1:num_u
                if      L2_urlos_FDB_US2      ~=
i2_urlos_FDB_US2
                sigma2_urlos_FDB_US2           =
sigma2_urlos_FDB_US2                           +
G2_urlos_FDB_US2(i2_urlos_FDB_US2,L2_urlos_FDB_U
S2)                                             *
wrf_urlos_FDB_US2(L2_urlos_FDB_US2,j2_urlos_FDB_
US2);
                end
            end

nij2_urlos_FDB_US2=sigma2_urlos_FDB_US2;
            if nij2_urlos_FDB_US2 == 0

wrf_urlos_FDB_US2(i2_urlos_FDB_US2,j2_urlos_FDB_
US2) = 1;
            else

wrf_urlos_FDB_US2(i2_urlos_FDB_US2,j2_urlos_FDB_
US2)                                            =
nij2_urlos_FDB_US2/abs(nij2_urlos_FDB_US2);
            end
            %i2_urlos_FDB_US2 = i2_urlos + 1;
        end
      %j2_urlos_FDB_US2 = j2_urlos + 1;
    end
    j_urlos_FDB_US2                             =
wrf_urlos_FDB_US2'*H_FDB_urlos_us2*vrf_urlos_FDB
_US2*Vd_urlos_FDB_US2*Vd_urlos_FDB_US2'*vrf_urlo
s_FDB_US2'*H_FDB_urlos_us2'*wrf_urlos_FDB_US2   +
variance_sistem*wrf_urlos_FDB_US2'*wrf_urlos_FDB
_US2;
    J_urlos_FDB_US2                             =
inv(j_urlos_FDB_US2);
    Wd_urlos_FDB_US2                            =
```

```
J_urlos_FDB_US2*wrf_urlos_FDB_US2'*H_FDB_urlos_u
s2*vrf_urlos_FDB_US2*Vd_urlos_FDB_US2;
    Wt_urlos_FDB_US2                                =
wrf_urlos_FDB_US2*Wd_urlos_FDB_US2;

    vrf_old52 = vrf_new52;
    vrf_new52                                       =
trace(wrf_urlos_FDB_US2*Wd_urlos_FDB_US2*Wd_urlo
s_FDB_US2'*wrf_urlos_FDB_US2');
    tol52 = vrf_new52 - vrf_old52;

jt52 = jt52 + 1;
end

SE_urlos_FDB_US2(SNRLoop,n_sistem)                  =
real(log2(det((eye(num_u))                          +
(1/variance_sistem)*Wt_urlos_FDB_US2*(inv(Wt_url
os_FDB_US2'*Wt_urlos_FDB_US2))*Wt_urlos_FDB_US2'
*H_FDB_urlos_us2*Vt_urlos_FDB_US2*Vt_urlos_FDB_U
S2'*H_FDB_urlos_us2')));

SE_urlos_FDBus2(SNRLoop,n_sistem)                   =
(SE_urlos_FDB_US1(SNRLoop,n_sistem)                 +
SE_urlos_FDB_US2(SNRLoop,n_sistem))/2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% MEMBUAT SISTEM DENGAN HYBRID BEAMFORMING
UNTUK USER 1

H_HB_urlos_us1      = H_urlos_us2(1:num_u,:);
F1_urlos_HB_US1                                     =
H_HB_urlos_us1'*H_HB_urlos_us1;
gamma_urlos_HB_US1                                  =
sqrt(P_sistem/(num_m*NRFt_HB_US1));
vrf_urlos_HB_US1     = ones(num_m,NRFt_HB_US1); %
step 1
l1_urlos_HB_US1     = length(num_m);
nij_urlos_HB_US1    = zeros(num_m,NRFt_HB_US1);
vrf_urlos1_HB_US1   = zeros(num_m,NRFt_HB_US1);
```

```
vrf_new53 = 100;
tol53 = 1;
jt53 = 0;

while (tol53 > 1e-5)
    for j1_urlos_HB_US1 = 1:1:NRFt_HB_US1 % step
2
        vrfj_urlos_HB_US1 = vrf_urlos_HB_US1;
        vrfj_urlos_HB_US1(:,j1_urlos_HB_US1)    =
[];
        I1_urlos_HB_US1 = eye(NRFt_HB_US1-1);
        C1_urlos_HB_US1   =   I1_urlos_HB_US1   +
((gamma_urlos_HB_US1^2/variance_sistem)*vrfj_url
os_HB_US1'*F1_urlos_HB_US1*vrfj_urlos_HB_US1);  %
step 3
        G1_urlos_HB_US1                          =
((gamma_urlos_HB_US1^2/variance_sistem)*F1_urlos
_HB_US1)                                         -
((gamma_urlos_HB_US1^4/variance_sistem^2)*F1_url
os_HB_US1*vrfj_urlos_HB_US1*(C1_urlos_HB_US1^(-
1))*vrfj_urlos_HB_US1'*F1_urlos_HB_US1); % step 4
        sigma_HB_urlos_US1 = 0;
        for i1_urlos_HB_US1 = 1:1:num_m % step 5
            for L1_urlos_HB_US1 = 1:1:num_m
                if       L1_urlos_HB_US1       ~=
i1_urlos_HB_US1
                    sigma_HB_urlos_US1          =
sigma_HB_urlos_US1                               +
G1_urlos_HB_US1(i1_urlos_HB_US1,L1_urlos_HB_US1)
*
vrf_urlos_HB_US1(L1_urlos_HB_US1,j1_urlos_HB_US1
); % step 6
                end
            end
            nij_urlos_HB_US1=sigma_HB_urlos_US1;
            if nij_urlos_HB_US1 == 0 % step 7

vrf_urlos_HB_US1(i1_urlos_HB_US1,j1_urlos_HB_US1
) = 1;
            else
```

```
vrf_urlos_HB_US1(i1_urlos_HB_US1,j1_urlos_HB_US1
) = nij_urlos_HB_US1/abs(nij_urlos_HB_US1);
            end
        end
        cekv_HB_US2 = abs(vrf_urlos_HB_US1);
    end
    I2_urlos_HB_US1 = eye(NRFr_HB_US1);
    VdVdH_HB_us1_HB_US1                          =
(sqrt(gamma_urlos_HB_US1^2))*I2_urlos_HB_US1;
    Heff_urlos_HB_US1                            =
H_HB_urlos_us1*vrf_urlos_HB_US1;
    Q_urlos_HB_US1                               =
vrf_urlos_HB_US1'*vrf_urlos_HB_US1;
    svd_urlos_HB_US1                             =
Heff_urlos_HB_US1*(Q_urlos_HB_US1^(-0.5));

[U_urlos_HB_US1,S_urlos_HB_US1,V_urlos_HB_US1]  =
svd(svd_urlos_HB_US1);
    Ue_urlos_HB_US1                              =
V_urlos_HB_US1(1:Ns_HB_US1,1:Ns_HB_US1);
    Te_urlos_HB_US1                              =
sqrt(P_sistem/NRFt_HB_US1)*(eye(Ns_HB_US1));
    Vd_urlos_HB_US1     =     (Q_urlos_HB_US1^(-
0.5))*Ue_urlos_HB_US1*Te_urlos_HB_US1;

    vrf_old53 = vrf_new53;
    vrf_new53                                    =
trace(vrf_urlos_HB_US1*Vd_urlos_HB_US1*Vd_urlos_
HB_US1'*vrf_urlos_HB_US1');
    tol53 = vrf_new53 - vrf_old53;
    jt53 = jt53 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO
Vt_urlos_HB_US1                                  =
vrf_urlos_HB_US1*Vd_urlos_HB_US1;
F2_urlos_HB_US1                                  =
H_HB_urlos_us1*Vt_urlos_HB_US1*Vt_urlos_HB_US1'*
```

```
H_HB_urlos_us1';

wrf_urlos_HB_US1 = ones(num_u,NRFr_HB_US1); %step
1
l2_urlos_HB_US1 = length(num_u);
nij2_urlos_HB_US1 = zeros(num_u,NRFr_HB_US1);
Wrf_urlos1_HB_US1 = zeros(num_u,NRFr_HB_US1);
vrf_new54 = 100;
tol54 = 1;
jt54 = 0;

while (tol54 > 1e-5)
    for j2_urlos_HB_US1 = 1:1:NRFr_HB_US1 %step 2
        wrfj_urlos_HB_US1 = wrf_urlos_HB_US1;
        wrfj_urlos_HB_US1(:,j2_urlos_HB_US1)    =
[];
        I2_urlos_HB_US1     =     eye(NRFr_HB_US1-
1); %step 3&4
        C2_urlos_HB_US1   =   I2_urlos_HB_US1   +
(((1/num_u)/variance_sistem)*wrfj_urlos_HB_US1'*
F2_urlos_HB_US1*wrfj_urlos_HB_US1);
        G2_urlos_HB_US1                          =
(((1/num_u)/variance_sistem)*F2_urlos_HB_US1)   -
(((1/num_u)^2/variance_sistem^2)*F2_urlos_HB_US1
*wrfj_urlos_HB_US1*(C2_urlos_HB_US1^(-
1))*wrfj_urlos_HB_US1'*F2_urlos_HB_US1);
        sigma2_HB_urlos_US1 = 0;
        for i2_urlos_HB_US1 = 1:1:num_u
            for L2_urlos_HB_US1 = 1:1:num_u
                if       L2_urlos_HB_US1       ~=
i2_urlos_HB_US1
                    sigma2_HB_urlos_US1         =
sigma2_HB_urlos_US1                             +
G2_urlos_HB_US1(i2_urlos_HB_US1,L2_urlos_HB_US1)
*
wrf_urlos_HB_US1(L2_urlos_HB_US1,j2_urlos_HB_US1
);
                end
            end
```

```
nij2_urlos_HB_US1=sigma2_HB_urlos_US1;
            if nij2_urlos_HB_US1 == 0

wrf_urlos_HB_US1(i2_urlos_HB_US1,j2_urlos_HB_US1
) = 1;
            else

wrf_urlos_HB_US1(i2_urlos_HB_US1,j2_urlos_HB_US1
) = nij2_urlos_HB_US1/abs(nij2_urlos_HB_US1);
            end
            %i2_urlos_HB_US1 = i2_urlos + 1;
        end
        %j2_urlos_HB_US1 = j2_urlos + 1;
    end
    j_urlos_HB_US1                                  =
wrf_urlos_HB_US1'*H_HB_urlos_us1*vrf_urlos_HB_US
1*Vd_urlos_HB_US1*Vd_urlos_HB_US1'*vrf_urlos_HB_
US1'*H_HB_urlos_us1'*wrf_urlos_HB_US1           +
variance_sistem*wrf_urlos_HB_US1'*wrf_urlos_HB_U
S1;
    J_urlos_HB_US1 = inv(j_urlos_HB_US1);
    Wd_urlos_HB_US1                                 =
J_urlos_HB_US1*wrf_urlos_HB_US1'*H_HB_urlos_us1*
vrf_urlos_HB_US1*Vd_urlos_HB_US1;
    Wt_urlos_HB_US1                                 =
wrf_urlos_HB_US1*Wd_urlos_HB_US1;

    vrf_old54 = vrf_new54;
    vrf_new54                                       =
trace(wrf_urlos_HB_US1*Wd_urlos_HB_US1*Wd_urlos_
HB_US1'*wrf_urlos_HB_US1');
    tol54 = vrf_new54 - vrf_old54;

    jt54 = jt54 + 1;
end

SE_urlos_HB_US1(SNRLoop,n_sistem)                   =
real(log2(det((eye(num_u))                          +
(1/variance_sistem)*Wt_urlos_HB_US1*(inv(Wt_urlo
s_HB_US1'*Wt_urlos_HB_US1))*Wt_urlos_HB_US1'*H_H
```

```
B_urlos_us1*Vt_urlos_HB_US1*Vt_urlos_HB_US1'*H_H
B_urlos_us1')));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%   MEMBUAT  SISTEM  HYBRID  BEAMFORMING  UNTUK
USER 2

H_HB_urlos_us2                                  =
H_urlos_us2((num_u+1):total_Rx_US2,:);
F1_urlos_HB_US2 = H_HB_urlos_us2'*H_HB_urlos_us2;
gamma_urlos_HB_US2                              =
sqrt(P_sistem/(num_m*NRFt_HB_US2));
vrf_urlos_HB_US2  =  ones(num_m,NRFt_HB_US2);  %
step 1
l1_urlos_HB_US2 = length(num_m);
nij_urlos_HB_US2 = zeros(num_m,NRFt_HB_US2);
vrf_urlos1_HB_US2 = zeros(num_m,NRFt_HB_US2);
vrf_new55 = 100;
tol55 = 1;
jt55 = 0;

while (tol55 > 1e-5)
    for j1_urlos_HB_US2 = 1:1:NRFt_HB_US2 % step
2
        vrfj_urlos_HB_US2 = vrf_urlos_HB_US2;
        vrfj_urlos_HB_US2(:,j1_urlos_HB_US2)    =
[];
        I1_urlos_HB_US2 = eye(NRFt_HB_US2-1);
        C1_urlos_HB_US2   =   I1_urlos_HB_US2   +
((gamma_urlos_HB_US2^2/variance_sistem)*vrfj_url
os_HB_US2'*F1_urlos_HB_US2*vrfj_urlos_HB_US2);  %
step 3
        G1_urlos_HB_US2                         =
((gamma_urlos_HB_US2^2/variance_sistem)*F1_urlos
_HB_US2)                                        -
((gamma_urlos_HB_US2^4/variance_sistem^2)*F1_url
os_HB_US2*vrfj_urlos_HB_US2*(C1_urlos_HB_US2^(-
1))*vrfj_urlos_HB_US2'*F1_urlos_HB_US2); % step 4
        sigma_HB_urlos_US2 = 0;
```

```
        for i1_urlos_HB_US2 = 1:1:num_m % step 5
            %sigma_HB_US2 = 0;
            for L1_urlos_HB_US2 = 1:1:num_m
                %sigma_HB_US2 = 0;
                if       L1_urlos_HB_US2       ~=
i1_urlos_HB_US2
                    sigma_HB_urlos_US2          =
sigma_HB_urlos_US2                              +
G1_urlos_HB_US2(i1_urlos_HB_US2,L1_urlos_HB_US2)
*
vrf_urlos_HB_US2(L1_urlos_HB_US2,j1_urlos_HB_US2
); % step 6
                end
            end
            nij_urlos_HB_US2=sigma_HB_urlos_US2;
            if nij_urlos_HB_US2 == 0 % step 7

vrf_urlos_HB_US2(i1_urlos_HB_US2,j1_urlos_HB_US2
) = 1;
            else

vrf_urlos_HB_US2(i1_urlos_HB_US2,j1_urlos_HB_US2
) = nij_urlos_HB_US2/abs(nij_urlos_HB_US2);
            end
        end
        cekv_HB_US2 = abs(vrf_urlos_HB_US2);
    end
    I2_urlos_HB_US2 = eye(NRFr_HB_US2);
    VdVdH_HB_us2_HB_US2                         =
(sqrt(gamma_urlos_HB_US2^2))*I2_urlos_HB_US2;
    Heff_urlos_HB_US2                           =
H_HB_urlos_us2*vrf_urlos_HB_US2;
    Q_urlos_HB_US2                              =
vrf_urlos_HB_US2'*vrf_urlos_HB_US2;
    svd_urlos_HB_US2                            =
Heff_urlos_HB_US2*(Q_urlos_HB_US2^(-0.5));

[U_urlos_HB_US2,S_urlos_HB_US2,V_urlos_HB_US2]  =
svd(svd_urlos_HB_US2);
    Ue_urlos_HB_US2                             =
```

```
V_urlos_HB_US2(1:Ns_HB_US2,1:Ns_HB_US2);
    Te_urlos_HB_US2                                    =
sqrt(P_sistem/NRFt_HB_US2)*(eye(Ns_HB_US2));
    Vd_urlos_HB_US2      =      (Q_urlos_HB_US2^(-
0.5))*Ue_urlos_HB_US2*Te_urlos_HB_US2;

    vrf_old55 = vrf_new55;
    vrf_new55                                          =
trace(vrf_urlos_HB_US2*Vd_urlos_HB_US2*Vd_urlos_
HB_US2'*vrf_urlos_HB_US2');
    tol55 = vrf_new55 - vrf_old55;

    jt55 = jt55 + 1;
end

% Algoritma 2 : Desain Hybrid Beamformers untuk
point-to-point MIMO
Vt_urlos_HB_US2                                        =
vrf_urlos_HB_US2*Vd_urlos_HB_US2;
F2_urlos_HB_US2                                        =
H_HB_urlos_us2*Vt_urlos_HB_US2*Vt_urlos_HB_US2'*
H_HB_urlos_us2';

wrf_urlos_HB_US2 = ones(num_u,NRFr_HB_US2); %step
1
l2_urlos_HB_US2 = length(num_u);
nij2_urlos_HB_US2 = zeros(num_u,NRFr_HB_US2);
Wrf_urlos1_HB_US2 = zeros(num_u,NRFr_HB_US2);
vrf_new56 = 100;
tol56 = 1;
jt56 = 0;

while (tol56 > 1e-5)
    for j2_urlos_HB_US2 = 1:1:NRFr_HB_US2 %step 2
        wrfj_urlos_HB_US2 = wrf_urlos_HB_US2;
        wrfj_urlos_HB_US2(:,j2_urlos_HB_US2)    =
[];
        I2_urlos_HB_US2     =     eye(NRFr_HB_US2-
1); %step 3&4
        C2_urlos_HB_US2    =    I2_urlos_HB_US2    +
```

```
(((1/num_u)/variance_sistem)*wrfj_urlos_HB_US2'*
F2_urlos_HB_US2*wrfj_urlos_HB_US2);
        G2_urlos_HB_US2                                =
(((1/num_u)/variance_sistem)*F2_urlos_HB_US2)   -
(((1/num_u)^2/variance_sistem^2)*F2_urlos_HB_US2
*wrfj_urlos_HB_US2*(C2_urlos_HB_US2^(-
1))*wrfj_urlos_HB_US2'*F2_urlos_HB_US2);
        sigma2_urlos_HB_US2 = 0;
        for i2_urlos_HB_US2 = 1:1:num_u
            for L2_urlos_HB_US2 = 1:1:num_u
                if       L2_urlos_HB_US2       ~=
i2_urlos_HB_US2
                    sigma2_urlos_HB_US2         =
sigma2_urlos_HB_US2                             +
G2_urlos_HB_US2(i2_urlos_HB_US2,L2_urlos_HB_US2)
*
wrf_urlos_HB_US2(L2_urlos_HB_US2,j2_urlos_HB_US2
);
                end
            end

nij2_urlos_HB_US2=sigma2_urlos_HB_US2;
            if nij2_urlos_HB_US2 == 0

wrf_urlos_HB_US2(i2_urlos_HB_US2,j2_urlos_HB_US2
) = 1;
            else

wrf_urlos_HB_US2(i2_urlos_HB_US2,j2_urlos_HB_US2
) = nij2_urlos_HB_US2/abs(nij2_urlos_HB_US2);
            end
            %i2_urlos_HB_US2 = i2_urlos + 1;
        end
        %j2_urlos_HB_US2 = j2_urlos + 1;
    end
    j_urlos_HB_US2                                 =
wrf_urlos_HB_US2'*H_HB_urlos_us2*vrf_urlos_HB_US
2*Vd_urlos_HB_US2*Vd_urlos_HB_US2'*vrf_urlos_HB_
US2'*H_HB_urlos_us2'*wrf_urlos_HB_US2           +
variance_sistem*wrf_urlos_HB_US2'*wrf_urlos_HB_U
```

```
S2;
    J_urlos_HB_US2 = inv(j_urlos_HB_US2);
    Wd_urlos_HB_US2                                =
J_urlos_HB_US2*wrf_urlos_HB_US2'*H_HB_urlos_us2*
vrf_urlos_HB_US2*Vd_urlos_HB_US2;
    Wt_urlos_HB_US2                                =
wrf_urlos_HB_US2*Wd_urlos_HB_US2;

    vrf_old56 = vrf_new56;
    vrf_new56                                      =
trace(wrf_urlos_HB_US2*Wd_urlos_HB_US2*Wd_urlos_
HB_US2'*wrf_urlos_HB_US2');
    tol56 = vrf_new56 - vrf_old56;

    jt56 = jt56 + 1;
end

SE_urlos_HB_US2(SNRLoop,n_sistem)                  =
real(log2(det((eye(num_u))                         +
(1/variance_sistem)*Wt_urlos_HB_US2*(inv(Wt_urlo
s_HB_US2'*Wt_urlos_HB_US2))*Wt_urlos_HB_US2'*H_H
B_urlos_us2*Vt_urlos_HB_US2*Vt_urlos_HB_US2'*H_H
B_urlos_us2')));

SE_urlos_HBus2(SNRLoop,n_sistem)                   =
(SE_urlos_HB_US1(SNRLoop,n_sistem)                 +
SE_urlos_HB_US2(SNRLoop,n_sistem))/2;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% MENCOBA MENCARI BER DARI BPSK SISTEM HYBRID
BEAMFORMING POINT-TO-POINT MIMO

%%%%  TES  DATA  STREAM  DI  TRANSMITTER  SAMPAI
RECEIVER
    sample_data = 1000;
    for stream = 1:1:sample_data
    N_data = Ns_HB;
    ip(:,stream)  =  rand(N_data,1)>0.5;        %
```

```
generating 0,1 with equal probability
    s_TX(:,stream)        =        2*ip(:,stream)-
1;                  % BPSK modulation 0 -> -1; 1
-> 1

    %%%% KANAL IRF
    %%%%%  PENGECEKAN  BER  UNTUK  FULL-DIGITAL
BEAMFORMING
    fase1_irf_FDB(:,stream)                      =
Vt_irf_FDB*s_TX(:,stream);          % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_FDB(:,stream)                      =
H_irf*fase1_irf_FDB(:,stream);     % Hasil olahan
fase 1 kemudian melewati kanal
    fase3_irf_FDB(:,stream)                      =
Wt_irf_FDB'*fase2_irf_FDB(:,stream);      % Hasil
sinyal  setelah  melewati  kanal,  diolah  oleh
penerima

    ipHat_IRF_FDB(:,stream)                      =
real(fase3_irf_FDB(:,stream))>0;
    for i_sinyal_irf_FDB = 1:Ns_FDB
        if
ipHat_IRF_FDB(i_sinyal_irf_FDB,stream) == 0

sinyal_rx_irf_FDB(i_sinyal_irf_FDB,stream) = -1;
        else
sinyal_rx_irf_FDB(i_sinyal_irf_FDB,stream) = 1;
        end
    end
    nErr_IRF_FDB(n_sistem,SNRLoop)               =
size(find([ip- ipHat_IRF_FDB]),1);
    nErr_IRF_FDB(nErr_IRF_FDB==0) = 1;

    %%%%% PENGECEKAN BER UNTUK HYBRID BEAMFORMING
    fase1_irf_HB(:,stream)                       =
Vt_irf_HB*s_TX(:,stream);           % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_HB(:,stream)                       =
H_irf*fase1_irf_HB(:,stream);      % Hasil olahan
```

```
fase 1 kemudian melewati kanal
    fase3_irf_HB(:,stream)                          =
Wt_irf_HB'*fase2_irf_HB(:,stream);       % Hasil
sinyal  setelah  melewati  kanal,  diolah  oleh
penerima

    ipHat_irf_HB(:,stream)                          =
real(fase3_irf_HB(:,stream))>0;
    for i_sinyal_irf_HB = 1:Ns_HB
        if   ipHat_irf_HB(i_sinyal_irf_HB,stream)
== 0

sinyal_rx_irf_HB(i_sinyal_irf_HB,stream) = -1;
        else
sinyal_rx_irf_HB(i_sinyal_irf_HB,stream) = 1;
        end
    end
    nErr_IRF_HB(n_sistem,SNRLoop)                   =
size(find([ip- ipHat_irf_HB]),1);
    nErr_IRF_HB(nErr_IRF_HB==0) = 1;

    %%% KANAL URLOS
    %%%   PENGECEKAN   BER   UNTUK   FULL-DIGITAL
BEAMFORMING
    fase1_urlos_FDB(:,stream)                       =
Vt_urlos_FDB*s_TX(:,stream);        % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_urlos_FDB(:,stream)                       =
H_urlos*fase1_urlos_FDB(:,stream);       % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_urlos_FDB(:,stream)                       =
Wt_urlos_FDB'*fase2_urlos_FDB(:,stream);       %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_FDB(:,stream)                       =
real(fase3_urlos_FDB(:,stream))>0;
    for i_sinyal_urlos_FDB = 1:Ns_FDB
        if
ipHat_urlos_FDB(i_sinyal_urlos_FDB,stream) == 0
```

```
sinyal_rx_urlos_FDB(i_sinyal_urlos_FDB,stream)  =
-1;
        else
sinyal_rx_urlos_FDB(i_sinyal_urlos_FDB,stream)  =
1;
        end
    end
    nErr_URLOS_FDB(n_sistem,SNRLoop)            =
size(find([ip- ipHat_urlos_FDB]),1);
    nErr_URLOS_FDB(nErr_URLOS_FDB==0) = 1;

    %%%%% PENGECEKAN BER UNTUK HYBRID BEAMFORMING
    fase1_urlos_HB(:,stream)                    =
Vt_urlos_HB*s_TX(:,stream);         % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_urlos_HB(:,stream)                    =
H_urlos*fase1_urlos_HB(:,stream);        % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_urlos_HB(:,stream)                    =
Wt_urlos_HB'*fase2_urlos_HB(:,stream);    % Hasil
sinyal  setelah  melewati  kanal,  diolah  oleh
penerima

    ipHat_urlos_HB(:,stream)                    =
real(fase3_urlos_HB(:,stream))>0;
    for i_sinyal_urlos_HB = 1:Ns_HB
        if
ipHat_urlos_HB(i_sinyal_urlos_HB,stream) == 0

sinyal_rx_urlos_HB(i_sinyal_urlos_HB,stream)  = -
1;
        else
sinyal_rx_urlos_HB(i_sinyal_urlos_HB,stream) = 1;
        end
    end
    nErr_URLOS_HB(n_sistem,SNRLoop)             =
size(find([ip- ipHat_irf_HB]),1);
    nErr_URLOS_HB(nErr_URLOS_HB==0) = 1;
```

```
    %%%%%%% TX = 64 dengan kanal IRF
    fase1_IRF_TX64(:,stream)                    =
Vt_irf_TX64*s_TX(:,stream);         % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_IRF_TX64(:,stream)                    =
H_irf*fase1_IRF_TX64(:,stream);          % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_IRF_TX64(:,stream)                    =
Wt_irf_TX64'*fase2_IRF_TX64(:,stream);    % Hasil
sinyal  setelah  melewati  kanal,  diolah  oleh
penerima

    ipHat_irf_TX64(:,stream)                    =
real(fase3_IRF_TX64(:,stream))>0;
    for i_sinyal_irf_TX64 = 1:Ns_HB
        if
ipHat_irf_TX64(i_sinyal_irf_TX64,stream) == 0

sinyal_rx_irf_TX64(i_sinyal_irf_TX64,stream) = -
1;
        else
sinyal_rx_irf_TX64(i_sinyal_irf_TX64,stream) = 1;
        end
    end
    nErr_irf_TX64(n_sistem,SNRLoop)             =
size(find([ip- ipHat_irf_TX64]),1);
    nErr_irf_TX64(nErr_irf_TX64==0) = 1;

    %%%%%%%% TX = 32 dengan kanal IRF
    fase1_irf_TX32(:,stream)                    =
Vt_irf_TX32*s_TX(:,stream);         % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_TX32(:,stream)                    =
H_irf_TX32*fase1_irf_TX32(:,stream);      % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_irf_TX32(:,stream)                    =
Wt_irf_TX32'*fase2_irf_TX32(:,stream);    % Hasil
sinyal  setelah  melewati  kanal,  diolah  oleh
penerima
```

```
    ipHat_irf_TX32(:,stream)                    =
real(fase3_irf_TX32(:,stream))>0;
    for i_sinyal_irf_TX32 = 1:Ns_HB
        if
ipHat_irf_TX32(i_sinyal_irf_TX32,stream) == 0

sinyal_rx_irf_TX32(i_sinyal_irf_TX32,stream) = -
1;
        else
sinyal_rx_irf_TX32(i_sinyal_irf_TX32,stream) = 1;
        end
    end
    nErr_irf_TX32(n_sistem,SNRLoop)             =
size(find([ip- ipHat_irf_TX32]),1);
    nErr_irf_TX32(nErr_irf_TX32==0) = 1;

    %%%%%%%% TX = 16 dengan kanal IRF
    fase1_irf_TX16(:,stream)                    =
Vt_irf_TX16*s_TX(:,stream);         % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_TX16(:,stream)                    =
H_irf_TX16*fase1_irf_TX16(:,stream);      % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_irf_TX16(:,stream)                    =
Wt_irf_TX16'*fase2_irf_TX16(:,stream);    % Hasil
sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_irf_TX16(:,stream)                    =
real(fase3_irf_TX16(:,stream))>0;
    for i_sinyal_irf_TX16 = 1:Ns_HB
        if
ipHat_irf_TX16(i_sinyal_irf_TX16,stream) == 0

sinyal_rx_irf_TX16(i_sinyal_irf_TX16,stream) = -
1;
        else
sinyal_rx_irf_TX16(i_sinyal_irf_TX16,stream) = 1;
        end
    end
```

```
    nErr_irf_TX16(n_sistem,SNRLoop)             =
size(find([ip- ipHat_irf_TX16]),1);
    nErr_irf_TX16(nErr_irf_TX16==0) = 1;

    %%%%%%% TX = 64 pada kanal URLOS
    fase1_urlos_TX64(:,stream)                  =
Vt_urlos_TX64*s_TX(:,stream);              % Data
stream diolah oleh tx untuk kemudian dipancarkan
    fase2_urlos_TX64(:,stream)                  =
H_urlos*fase1_urlos_TX64(:,stream);       % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_urlos_TX64(:,stream)                  =
Wt_urlos_TX64'*fase2_urlos_TX64(:,stream);      %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_TX64(:,stream)                  =
real(fase3_urlos_TX64(:,stream))>0;
    for i_sinyal_urlos_TX64 = 1:Ns_HB
        if
ipHat_urlos_TX64(i_sinyal_urlos_TX64,stream) == 0

sinyal_rx_urlos_TX64(i_sinyal_urlos_TX64,stream)
= -1;
        else
sinyal_rx_urlos_TX64(i_sinyal_urlos_TX64,stream)
= 1;
        end
    end
    nErr_urlos_TX64(n_sistem,SNRLoop)           =
size(find([ip- ipHat_urlos_TX64]),1);
    nErr_urlos_TX64(nErr_urlos_TX64==0) = 1;

    %%%%%%% TX = 32 pada kanal URLOS
    fase1_urlos_TX32(:,stream)                  =
Vt_urlos_TX32*s_TX(:,stream);              % Data
stream diolah oleh tx untuk kemudian dipancarkan
    fase2_urlos_TX32(:,stream)                  =
H_urlos_TX32*fase1_urlos_TX32(:,stream);        %
Hasil olahan fase 1 kemudian melewati kanal
```

```
    fase3_urlos_TX32(:,stream)                  =
Wt_urlos_TX32'*fase2_urlos_TX32(:,stream);      %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_TX32(:,stream)                  =
real(fase3_urlos_TX32(:,stream))>0;
    for i_sinyal_urlos_TX32 = 1:Ns_HB
        if
ipHat_urlos_TX32(i_sinyal_urlos_TX32,stream) == 0

sinyal_rx_urlos_TX32(i_sinyal_urlos_TX32,stream)
= -1;
        else
sinyal_rx_urlos_TX32(i_sinyal_urlos_TX32,stream)
= 1;
        end
    end
    nErr_urlos_TX32(n_sistem,SNRLoop)           =
size(find([ip- ipHat_urlos_TX32]),1);
    nErr_urlos_TX32(nErr_urlos_TX32==0) = 1;

    %%%%%% TX = 16 pada kanal URLOS
    fase1_urlos_TX16(:,stream)                  =
Vt_urlos_TX16*s_TX(:,stream);               % Data
stream diolah oleh tx untuk kemudian dipancarkan
    fase2_urlos_TX16(:,stream)                  =
H_urlos_16*fase1_urlos_TX16(:,stream);          %
Hasil olahan fase 1 kemudian melewati kanal
    fase3_urlos_TX16(:,stream)                  =
Wt_urlos_TX16'*fase2_urlos_TX16(:,stream);      %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_TX16(:,stream)                  =
real(fase3_urlos_TX16(:,stream))>0;
    for i_sinyal_urlos_TX16 = 1:Ns_HB
        if
ipHat_urlos_TX16(i_sinyal_urlos_TX16,stream) == 0
```

```
sinyal_rx_urlos_TX16(i_sinyal_urlos_TX16,stream)
= -1;
        else
sinyal_rx_urlos_TX16(i_sinyal_urlos_TX16,stream)
= 1;
        end
    end
    nErr_urlos_TX16(n_sistem,SNRLoop)           =
size(find([ip- ipHat_urlos_TX16]),1);
    nErr_urlos_TX16(nErr_urlos_TX16==0) = 1;

    %%%%% PENGECEKAN BER UNTUK RX = 64 kanal IRF
    fase1_irf_RX64(:,stream)                    =
Vt_irf_RX64*s_TX(:,stream);         % Data stream
diolah oleh RX untuk kemudian dipancarkan
    fase2_irf_RX64(:,stream)                    =
H_irf*fase1_irf_RX64(:,stream);          % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_irf_RX64(:,stream)                    =
Wt_irf_RX64'*fase2_irf_RX64(:,stream);    % Hasil
sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_irf_RX64(:,stream)                    =
real(fase3_irf_RX64(:,stream))>0;
    for i_sinyal_irf_RX64 = 1:Ns_HB
        if
ipHat_irf_RX64(i_sinyal_irf_RX64,stream) == 0

sinyal_rx_irf_RX64(i_sinyal_irf_RX64,stream) = -
1;
        else
sinyal_rx_irf_RX64(i_sinyal_irf_RX64,stream) = 1;
        end
    end
    nErr_irf_RX64(n_sistem,SNRLoop)             =
size(find([ip- ipHat_irf_RX64]),1);
    nErr_irf_RX64(nErr_irf_RX64==0) = 1;

    %%%%% PENGECEKAN BER UNTUK RX = 32 kanal IRF
```

```
    fase1_irf_RX32(:,stream)                    =
Vt_irf_RX32*s_TX(:,stream);         % Data stream
diolah oleh RX untuk kemudian dipancarkan
    fase2_irf_RX32(:,stream)                    =
H_irf_RX32*fase1_irf_RX32(:,stream);      % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_irf_RX32(:,stream)                    =
Wt_irf_RX32'*fase2_irf_RX32(:,stream);    % Hasil
sinyal  setelah  melewati  kanal,  diolah  oleh
penerima

    ipHat_irf_RX32(:,stream)                    =
real(fase3_irf_RX32(:,stream))>0;
    for i_sinyal_irf_RX32 = 1:Ns_HB
        if
ipHat_irf_RX32(i_sinyal_irf_RX32,stream) == 0

sinyal_rx_irf_RX32(i_sinyal_irf_RX32,stream)  = -
1;
        else
sinyal_rx_irf_RX32(i_sinyal_irf_RX32,stream) = 1;
        end
    end
    nErr_irf_RX32(n_sistem,SNRLoop)             =
size(find([ip- ipHat_irf_RX32]),1);
    nErr_irf_RX32(nErr_irf_RX32==0) = 1;

    %%%%% PENGECEKAN BER UNTUK RX = 16 kanal IRF
    fase1_irf_RX16(:,stream)                    =
Vt_irf_RX16*s_TX(:,stream);         % Data stream
diolah oleh RX untuk kemudian dipancarkan
    fase2_irf_RX16(:,stream)                    =
H_irf_RX16*fase1_irf_RX16(:,stream);      % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_irf_RX16(:,stream)                    =
Wt_irf_RX16'*fase2_irf_RX16(:,stream);    % Hasil
sinyal  setelah  melewati  kanal,  diolah  oleh
penerima

    ipHat_irf_RX16(:,stream)                    =
```

```
real(fase3_irf_RX16(:,stream))>0;
    for i_sinyal_irf_RX16 = 1:Ns_HB
        if
ipHat_irf_RX16(i_sinyal_irf_RX16,stream) == 0

sinyal_rx_irf_RX16(i_sinyal_irf_RX16,stream) = -
1;
        else
sinyal_rx_irf_RX16(i_sinyal_irf_RX16,stream) = 1;
        end
    end
    nErr_irf_RX16(n_sistem,SNRLoop)             =
size(find([ip- ipHat_irf_RX16]),1);
    nErr_irf_RX16(nErr_irf_RX16==0) = 1;

    %%%%% sistem dengan Rx = 16 pada kanal URLOS
    fase1_urlos_RX64(:,stream)                  =
Vt_urlos_RX64*s_TX(:,stream);              % Data
stream diolah oleh RX untuk kemudian dipancarkan
    fase2_urlos_RX64(:,stream)                  =
H_urlos*fase1_urlos_RX64(:,stream);       % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_urlos_RX64(:,stream)                  =
Wt_urlos_RX64'*fase2_urlos_RX64(:,stream);      %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_RX64(:,stream)                  =
real(fase3_urlos_RX64(:,stream))>0;
    for i_sinyal_urlos_RX64 = 1:Ns_HB
        if
ipHat_urlos_RX64(i_sinyal_urlos_RX64,stream) == 0

sinyal_rx_urlos_RX64(i_sinyal_urlos_RX64,stream)
= -1;
        else
sinyal_rx_urlos_RX64(i_sinyal_urlos_RX64,stream)
= 1;
        end
    end
```

```
    nErr_urlos_RX64(n_sistem,SNRLoop)           =
size(find([ip- ipHat_urlos_RX64]),1);
    nErr_urlos_RX64(nErr_urlos_RX64==0) = 1;

    %%%%% sistem dengan Rx = 12 pada kanal URLOS
    fase1_urlos_RX32(:,stream)                  =
Vt_urlos_RX32*s_TX(:,stream);              % Data
stream diolah oleh RX untuk kemudian dipancarkan
    fase2_urlos_RX32(:,stream)                  =
H_urlos_RX32*fase1_urlos_RX32(:,stream);        %
Hasil olahan fase 1 kemudian melewati kanal
    fase3_urlos_RX32(:,stream)                  =
Wt_urlos_RX32'*fase2_urlos_RX32(:,stream);      %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_RX32(:,stream)                  =
real(fase3_urlos_RX32(:,stream))>0;
    for i_sinyal_urlos_RX32 = 1:Ns_HB
        if
ipHat_urlos_RX32(i_sinyal_urlos_RX32,stream) == 0

sinyal_rx_urlos_RX32(i_sinyal_urlos_RX32,stream)
= -1;
        else
sinyal_rx_urlos_RX32(i_sinyal_urlos_RX32,stream)
= 1;
        end
    end
    nErr_urlos_RX32(n_sistem,SNRLoop)           =
size(find([ip- ipHat_urlos_RX32]),1);
    nErr_urlos_RX32(nErr_urlos_RX32==0) = 1;

    %%%%% sistem dengan Rx = 8 pada kanal URLOS
    fase1_urlos_RX16(:,stream)                  =
Vt_urlos_RX16*s_TX(:,stream);              % Data
stream diolah oleh RX untuk kemudian dipancarkan
    fase2_urlos_RX16(:,stream)                  =
H_urlos_RX16*fase1_urlos_RX16(:,stream);        %
Hasil olahan fase 1 kemudian melewati kanal
```

```
    fase3_urlos_RX16(:,stream)                  =
Wt_urlos_RX16'*fase2_urlos_RX16(:,stream);      %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_RX16(:,stream)                  =
real(fase3_urlos_RX16(:,stream))>0;
    for i_sinyal_urlos_RX16 = 1:Ns_HB
        if
ipHat_urlos_RX16(i_sinyal_urlos_RX16,stream) == 0

sinyal_rx_urlos_RX16(i_sinyal_urlos_RX16,stream)
= -1;
        else
sinyal_rx_urlos_RX16(i_sinyal_urlos_RX16,stream)
= 1;
        end
    end
    nErr_urlos_RX16(n_sistem,SNRLoop)           =
size(find([ip- ipHat_urlos_RX16]),1);
    nErr_urlos_RX16(nErr_urlos_RX16==0) = 1;

    %%%%%%%%%%%%%% FDB untuk user 1
    fase1_irf_FDB_US1(:,stream)                 =
Vt_irf_FDB_US1*s_TX(:,stream);              % Data
stream diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_FDB_US1(:,stream)                 =
H_FDB_irf_us1*fase1_irf_FDB_US1(:,stream);      %
Hasil olahan fase 1 kemudian melewati kanal
    fase3_irf_FDB_US1(:,stream)                 =
Wt_irf_FDB_US1'*fase2_irf_FDB_US1(:,stream);    %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_irf_FDB_US1(:,stream)                 =
real(fase3_irf_FDB_US1(:,stream))>0;
    for i_sinyal_irf_FDB_US1 = 1:Ns_FDB_US1
        if
ipHat_irf_FDB_US1(i_sinyal_irf_FDB_US1,stream)
== 0
```

```
sinyal_rx_irf_FDB_US1(i_sinyal_irf_FDB_US1,strea
m) = -1;
        else
sinyal_rx_irf_FDB_US1(i_sinyal_irf_FDB_US1,strea
m) = 1;
        end
    end
    nErr_FDB_irf_US1(n_sistem,SNRLoop)          =
size(find([ip- ipHat_irf_FDB_US1]),1);
    nErr_FDB_irf_US1(nErr_FDB_irf_US1==0) = 1;

    %%%%%%%%%%%%% FDB untuk user 2
    fase1_irf_FDB_US2(:,stream)                 =
Vt_irf_FDB_US2*s_TX(:,stream);              % Data
stream diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_FDB_US2(:,stream)                 =
H_FDB_irf_us2*fase1_irf_FDB_US2(:,stream);      %
Hasil olahan fase 1 kemudian melewati kanal
    fase3_irf_FDB_US2(:,stream)                 =
Wt_irf_FDB_US2'*fase2_irf_FDB_US2(:,stream);    %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_irf_FDB_US2(:,stream)                 =
real(fase3_irf_FDB_US2(:,stream))>0;
    for i_sinyal_irf_FDB_US2 = 1:Ns_FDB_US2
        if
ipHat_irf_FDB_US2(i_sinyal_irf_FDB_US2,stream)
== 0

sinyal_rx_irf_FDB_US2(i_sinyal_irf_FDB_US2,strea
m) = -1;
        else
sinyal_rx_irf_FDB_US2(i_sinyal_irf_FDB_US2,strea
m) = 1;
        end
    end
    nErr_FDB_irf_US2(n_sistem,SNRLoop)          =
size(find([ip- ipHat_irf_FDB_US2]),1);
```

```
    nErr_FDB_irf_US2(nErr_FDB_irf_US2==0) = 1;

    %%%%%%%%%%%%%% HB untuk user 1
    fase1_irf_HB_US1(:,stream)                    =
Vt_irf_HB_US1*s_TX(:,stream);   %   Data   stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_HB_US1(:,stream)                    =
H_HB_irf_us1*fase1_irf_HB_US1(:,stream);        %
Hasil olahan fase 1 kemudian melewati kanal
    fase3_irf_HB_US1(:,stream)                    =
Wt_irf_HB_US1'*fase2_irf_HB_US1(:,stream);      %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_HB_irf_US1(:,stream)                    =
real(fase3_irf_HB_US1(:,stream))>0;
    for i_sinyal_irf_HB_US1 = 1:Ns_HB_US1
        if
ipHat_HB_irf_US1(i_sinyal_irf_HB_US1,stream) == 0

sinyal_rx_irf_HB_US1(i_sinyal_irf_HB_US1,stream)
= -1;
        else
sinyal_rx_irf_HB_US1(i_sinyal_irf_HB_US1,stream)
= 1;
        end
    end
    nErr_HB_irf_US1(n_sistem,SNRLoop)             =
size(find([ip- ipHat_HB_irf_US1]),1);
    nErr_HB_irf_US1(nErr_HB_irf_US1==0) = 1;

    %%%%%%%%%%%%%% HB untuk user 2
    fase1_irf_HB_US2(:,stream)                    =
Vt_irf_HB_US2*s_TX(:,stream);   %   Data   stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_HB_US2(:,stream)                    =
H_HB_irf_us2*fase1_irf_HB_US2(:,stream); % Hasil
olahan fase 1 kemudian melewati kanal
    fase3_irf_HB_US2(:,stream)                    =
Wt_irf_HB_US2'*fase2_irf_HB_US2(:,stream);      %
```

```
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_HB_irf_US2(:,stream)                    =
real(fase3_irf_HB_US2(:,stream))>0;
    for i_sinyal_irf_HB_US2 = 1:Ns_HB_US2
        if
ipHat_HB_irf_US2(i_sinyal_irf_HB_US2,stream) == 0

sinyal_rx_HB_irf_US2(i_sinyal_irf_HB_US2,stream)
= -1;
        else
sinyal_rx_HB_irf_US2(i_sinyal_irf_HB_US2,stream)
= 1;
        end
    end
    nErr_HB_irf_US2(n_sistem,SNRLoop)             =
size(find([ip- ipHat_HB_irf_US2]),1);
    nErr_HB_irf_US2(nErr_HB_irf_US2==0) = 1;

        %%%%%%%% FDB untuk user 1
    fase1_FDB_urlos_US1(:,stream)                 =
Vt_urlos_FDB_US1*s_TX(:,stream);           % Data
stream diolah oleh tx untuk kemudian dipancarkan
    fase2_FDB_urlos_US1(:,stream)                 =
H_FDB_urlos_us1*fase1_FDB_urlos_US1(:,stream);
% Hasil olahan fase 1 kemudian melewati kanal
    fase3_FDB_urlos_US1(:,stream)                 =
Wt_urlos_FDB_US1'*fase2_FDB_urlos_US1(:,stream);
% Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_FDB_urlos_US1(:,stream)                 =
real(fase3_FDB_urlos_US1(:,stream))>0;
    for i_sinyal_urlos_FDB_US1 = 1:Ns_FDB_US1
        if
ipHat_FDB_urlos_US1(i_sinyal_urlos_FDB_US1,strea
m) == 0

sinyal_rx_urlos_FDB_US1(i_sinyal_urlos_FDB_US1,s
```

```
tream) = -1;
        else
sinyal_rx_urlos_FDB_US1(i_sinyal_urlos_FDB_US1,s
tream) = 1;
        end
    end
    nErr_FDB_urlos_US1(n_sistem,SNRLoop)        =
size(find([ip- ipHat_FDB_urlos_US1]),1);
    nErr_FDB_urlos_US1(nErr_FDB_urlos_US1==0)   =
1;

    %%%%%%%%%%%%%%%% FDB untuk user 2
    fase1_FDB_urlos_US2(:,stream)               =
Vt_urlos_FDB_US2*s_TX(:,stream);           % Data
stream diolah oleh tx untuk kemudian dipancarkan
    fase2_FDB_urlos_US2(:,stream)               =
H_FDB_urlos_us2*fase1_FDB_urlos_US2(:,stream);
% Hasil olahan fase 1 kemudian melewati kanal
    fase3_FDB_urlos_US2(:,stream)               =
Wt_urlos_FDB_US2'*fase2_FDB_urlos_US2(:,stream);
% Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_FDB_urlos_US2(:,stream)               =
real(fase3_FDB_urlos_US2(:,stream))>0;
    for i_sinyal_urlos_FDB_US2 = 1:Ns_FDB_US2
        if
ipHat_FDB_urlos_US2(i_sinyal_urlos_FDB_US2,strea
m) == 0

sinyal_rx_FDB_urlos_US2(i_sinyal_urlos_FDB_US2,s
tream) = -1;
        else
sinyal_rx_FDB_urlos_US2(i_sinyal_urlos_FDB_US2,s
tream) = 1;
        end
    end
    nErr_FDB_urlos_US2(n_sistem,SNRLoop)        =
size(find([ip- ipHat_FDB_urlos_US2]),1);
    nErr_FDB_urlos_US2(nErr_FDB_urlos_US2==0)   =
```

```
1;

    %%%%%%%%%%%%%%%% HB untuk user 1
    fase1_HB_urlos_US1(:,stream)                =
Vt_urlos_HB_US1*s_TX(:,stream);  % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_HB_urlos_US1(:,stream)                =
H_HB_urlos_us1*fase1_HB_urlos_US1(:,stream);
% Hasil olahan fase 1 kemudian melewati kanal
    fase3_HB_urlos_US1(:,stream)                =
Wt_urlos_HB_US1'*fase2_HB_urlos_US1(:,stream);
% Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_HB_urlos_US1(:,stream)                =
real(fase3_HB_urlos_US1(:,stream))>0;
    for i_sinyal_urlos_HB_US1 = 1:Ns_HB_US1
        if
ipHat_HB_urlos_US1(i_sinyal_urlos_HB_US1,stream)
== 0

sinyal_rx_HB_urlos_US1(i_sinyal_urlos_HB_US1,str
eam) = -1;
        else
sinyal_rx_HB_urlos_US1(i_sinyal_urlos_HB_US1,str
eam) = 1;
        end
    end
    nErr_HB_urlos_US1(n_sistem,SNRLoop)         =
size(find([ip- ipHat_HB_urlos_US1]),1);
    nErr_HB_urlos_US1(nErr_HB_urlos_US1==0) = 1;

    %%%%%%%%%%%%%%%% HB untuk user 2
    fase1_HB_urlos_US2(:,stream)                =
Vt_urlos_HB_US2*s_TX(:,stream);  % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_HB_urlos_US2(:,stream)                =
H_HB_urlos_us2*fase1_HB_urlos_US2(:,stream);    %
Hasil olahan fase 1 kemudian melewati kanal
    fase3_HB_urlos_US2(:,stream)                =
```

```
Wt_urlos_HB_US2'*fase2_HB_urlos_US2(:,stream);  %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_HB_urlos_US2(:,stream)                =
real(fase3_HB_urlos_US2(:,stream))>0;
    for i_sinyal_HB_urlos_US2 = 1:Ns_HB_US2
        if
ipHat_HB_urlos_US2(i_sinyal_HB_urlos_US2,stream)
== 0

sinyal_rx_HB_urlos_US2(i_sinyal_HB_urlos_US2,str
eam) = -1;
        else
sinyal_rx_HB_urlos_US2(i_sinyal_HB_urlos_US2,str
eam) = 1;
        end
    end
    nErr_HB_urlos_US2(n_sistem,SNRLoop)         =
size(find([ip- ipHat_HB_urlos_US2]),1);
    nErr_HB_urlos_US2(nErr_HB_urlos_US2==0) = 1;

    end

    %%% RF Chain = 4
    %%%% RF chain = 4 pada kanal IRF
    sample_data_RF4 = 1500;
    for stream_RF4 = 1:1:sample_data_RF4
    N_data_RF4 = Ns_RF4;
    ip_RF4(:,stream_RF4)                        =
rand(N_data_RF4,1)>0.5;  %  generating  0,1  with
equal probability
    s_RF4(:,stream_RF4) = 2*ip_RF4(:,stream_RF4)-
1; % BPSK modulation 0 -> -1; 1 -> 1

    fase1_irf_RF4(:,stream_RF4)                 =
Vt_irf_RF4*s_RF4(:,stream_RF4);  %  Data  stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_irf_RF4(:,stream_RF4)                 =
H_irf*fase1_irf_RF4(:,stream_RF4);       % Hasil
```

```
olahan fase 1 kemudian melewati kanal
    fase3_irf_RF4(:,stream_RF4)                 =
Wt_irf_RF4'*fase2_irf_RF4(:,stream_RF4);        %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_irf_RF4(:,stream_RF4)                 =
real(fase3_irf_RF4(:,stream_RF4))>0;
    for i_sinyal_irf_RF4 = 1:Ns_RF4
        if
ipHat_irf_RF4(i_sinyal_irf_RF4,stream_RF4) == 0

sinyal_rx_irf_RF4(i_sinyal_irf_RF4,stream_RF4)  =
-1;
        else
sinyal_rx_irf_RF4(i_sinyal_irf_RF4,stream_RF4)  =
1;
        end
    end
    nErr_irf_RF4(n_sistem,SNRLoop)              =
size(find([ip_RF4- ipHat_irf_RF4]),1);
    nErr_irf_RF4(nErr_irf_RF4==0) = 1;

    %%%% RF chain = 4 pada kanal URLOS
    fase1_urlos_RF4(:,stream_RF4)               =
Vt_urlos_RF4*s_RF4(:,stream_RF4);  % Data stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_urlos_RF4(:,stream_RF4)               =
H_urlos*fase1_urlos_RF4(:,stream_RF4);   %  Hasil
olahan fase 1 kemudian melewati kanal
    fase3_urlos_RF4(:,stream_RF4)               =
Wt_urlos_RF4'*fase2_urlos_RF4(:,stream_RF4);    %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_RF4(:,stream_RF4)               =
real(fase3_urlos_RF4(:,stream_RF4))>0;
    for i_sinyal_urlos_RF4 = 1:Ns_RF4
        if
ipHat_urlos_RF4(i_sinyal_urlos_RF4,stream_RF4)
```

```
== 0

sinyal_rx_urlos_RF4(i_sinyal_urlos_RF4,stream_RF
4) = -1;
        else
sinyal_rx_urlos_RF4(i_sinyal_urlos_RF4,stream_RF
4) = 1;
        end
    end
    nErr_urlos_RF4(n_sistem,SNRLoop)            =
size(find([ip_RF4- ipHat_urlos_RF4]),1);
    nErr_urlos_RF4(nErr_urlos_RF4==0) = 1;
    end

    %%% RF Chain = 2
    %%%% RF chain = 2 pada kanal IRF
    sample_data_RF2 = 3000;
    for stream_RF2 = 1:1:sample_data_RF2
        N_data_RF2 = Ns_RF2;
        ip_RF2(:,stream_RF2)                    =
rand(N_data_RF2,1)>0.5;  % generating 0,1 with
equal probability
        s_RF2(:,stream_RF2)                     =
2*ip_RF2(:,stream_RF2)-1; % BPSK modulation 0 ->
-1; 1 -> 1

        fase1_irf_RF2(:,stream_RF2)             =
Vt_irf_RF2*s_RF2(:,stream_RF2);  % Data stream
diolah oleh tx untuk kemudian dipancarkan
        fase2_irf_RF2(:,stream_RF2)             =
H_irf*fase1_irf_RF2(:,stream_RF2); % Hasil olahan
fase 1 kemudian melewati kanal
        fase3_irf_RF2(:,stream_RF2)             =
Wt_irf_RF2'*fase2_irf_RF2(:,stream_RF2); % Hasil
sinyal setelah melewati kanal, diolah oleh
penerima

        ipHat_irf_RF2(:,stream_RF2)             =
real(fase3_irf_RF2(:,stream_RF2))>0;
        for i_sinyal_irf_RF2 = 1:Ns_RF2
```

```
            if
ipHat_irf_RF2(i_sinyal_irf_RF2,stream_RF2) == 0

sinyal_rx_irf_RF2(i_sinyal_irf_RF2,stream_RF2)  =
-1;
            else
sinyal_rx_irf_RF2(i_sinyal_irf_RF2,stream_RF2)  =
1;
            end
        end
    nErr_irf_RF2(n_sistem,SNRLoop)              =
size(find([ip_RF2- ipHat_irf_RF2]),1);
    nErr_irf_RF2(nErr_irf_RF2==0) = 1;

    %%%% RF chain = 2 pada kanal URLOS
    fase1_urlos_RF2(:,stream_RF2)               =
Vt_urlos_RF2*s_RF2(:,stream_RF2);  % Data  stream
diolah oleh tx untuk kemudian dipancarkan
    fase2_urlos_RF2(:,stream_RF2)               =
H_urlos*fase1_urlos_RF2(:,stream_RF2);   %   Hasil
olahan fase 1 kemudian melewati kanal
    fase3_urlos_RF2(:,stream_RF2)               =
Wt_urlos_RF2'*fase2_urlos_RF2(:,stream_RF2);    %
Hasil sinyal setelah melewati kanal, diolah oleh
penerima

    ipHat_urlos_RF2(:,stream_RF2)               =
real(fase3_urlos_RF2(:,stream_RF2))>0;
    for i_sinyal_urlos_RF2 = 1:Ns_RF2
        if
ipHat_urlos_RF2(i_sinyal_urlos_RF2,stream_RF2)
== 0

sinyal_rx_urlos_RF2(i_sinyal_urlos_RF2,stream_RF
2) = -1;
        else
sinyal_rx_urlos_RF2(i_sinyal_urlos_RF2,stream_RF
2) = 1;
        end
    end
```

```
    nErr_urlos_RF2(n_sistem,SNRLoop)            =
size(find([ip_RF2- ipHat_urlos_RF2]),1);
    nErr_urlos_RF2(nErr_urlos_RF2==0) = 1;

    end

end
end

simBER_IRF_FDB                                  =
nErr_IRF_FDB/(N_data*sample_data);
simBER_IRF_HB = nErr_IRF_HB/(N_data*sample_data);
simBER_URLOS_FDB                                =
nErr_URLOS_FDB/(N_data*sample_data);
simBER_URLOS_HB                                 =
nErr_URLOS_HB/(N_data*sample_data);

simBER_IRF_TX64                                 =
nErr_irf_TX64/(N_data*sample_data);
simBER_IRF_TX32                                 =
nErr_irf_TX32/(N_data*sample_data);
simBER_IRF_TX16                                 =
nErr_irf_TX16/(N_data*sample_data);

simBER_urlos_TX64                               =
nErr_urlos_TX64/(N_data*sample_data);
simBER_urlos_TX32                               =
nErr_urlos_TX32/(N_data*sample_data);
simBER_urlos_TX16                               =
nErr_urlos_TX16/(N_data*sample_data);

simBER_irf_RX64                                 =
nErr_irf_RX64/(N_data*sample_data);
simBER_irf_RX32                                 =
nErr_irf_RX32/(N_data*sample_data);
simBER_irf_RX16                                 =
nErr_irf_RX16/(N_data*sample_data);

simBER_urlos_RX64                               =
nErr_urlos_RX64/(N_data*sample_data);
```

```
simBER_urlos_RX32                                    =
nErr_urlos_RX32/(N_data*sample_data);
simBER_urlos_RX16                                    =
nErr_urlos_RX16/(N_data*sample_data);

simBER_irf_RF4                                       =
nErr_irf_RF4/(N_data*sample_data);
simBER_irf_RF2                                       =
nErr_irf_RF2/(N_data*sample_data);

simBER_urlos_RF4                                     =
nErr_urlos_RF4/(N_data_RF4*sample_data_RF4);
simBER_urlos_RF2                                     =
nErr_urlos_RF2/(N_data_RF2*sample_data_RF2);

simBER_FDB_irf_US1                                   =
nErr_FDB_irf_US1/(N_data*sample_data);
simBER_FDB_irf_US2                                   =
nErr_FDB_irf_US2/(N_data*sample_data);
simBER_irf_FDBus2    =    (simBER_FDB_irf_US1    +
simBER_FDB_irf_US2)/2;

simBER_HB_irf_US1                                    =
nErr_HB_irf_US1/(N_data*sample_data);
simBER_HB_irf_US2                                    =
nErr_HB_irf_US2/(N_data*sample_data);
simBER_irf_HBus2     =     (simBER_HB_irf_US1     +
simBER_HB_irf_US2)/2;

simBER_FDB_US1                                       =
nErr_FDB_urlos_US1/(N_data*sample_data);
simBER_FDB_US2                                       =
nErr_FDB_urlos_US2/(N_data*sample_data);
simBER_FDBus2       =       (simBER_FDB_US1       +
simBER_FDB_US2)/2;

simBER_HB_US1                                        =
nErr_HB_urlos_US1/(N_data*sample_data);
simBER_HB_US2                                        =
nErr_HB_urlos_US2/(N_data*sample_data);
```

```
simBER_HBus2 = (simBER_HB_US1 + simBER_HB_US2)/2;

%%% PLOT POLA RADIASI PADA OUTPUT TX
antena_urlos_Tx = sum(Vt_urlos_HB,2);
amplitudoAF_urlos_TX = abs(antena_urlos_Tx);
faseAf_rad_urlosTX = angle(antena_urlos_Tx);
faseAf_deg_urlosTX = rad2deg(faseAf_rad_urlosTX);
AF1_TX = zeros(num_m,360);

for Tx_urlos = 1:num_m
    beta_urlosTX = 2*pi;
    d_urlosTX = 1/2;
    N1_urlosTX = 64 ;
    i = sqrt(-1);
    An_urlosTX                                  =
amplitudoAF_urlos_TX(Tx_urlos,:);
    alpha_urlosTX                               =
faseAf_rad_urlosTX(Tx_urlos,:);
    for theta_urlosTX = 1 : 360
        rad(theta_urlosTX)                      =
(theta_urlosTX*pi)/180;
        for n_TXurlos = 1 : N1_urlosTX
            AF1_TX(Tx_urlos,theta_urlosTX)      =
AF1_TX(Tx_urlos,theta_urlosTX)                  +
An_urlosTX*exp(i*(n_TXurlos-1)*((beta_urlosTX   *
d_urlosTX   *   sin(rad(theta_urlosTX))   )   +
alpha_urlosTX));
        end
        AF1_TX(Tx_urlos,theta_urlosTX)          =
abs(AF1_TX(Tx_urlos,theta_urlosTX));
end
end
total_radiasiTx_irf = sum(AF1_TX,1);
radiasi_urlosTX                                 =
total_radiasiTx_irf/max(total_radiasiTx_irf);

%%% PLOT POLA RADIASI PADA OUTPUT RX
input_urlosRx                                   =
(H_urlos*Vt_urlos_HB)*wrf_urlos_HB';
antena_urlosRX = (sum(input_urlosRx,2));
```

```
amplitudoAF_urlosRX = abs(antena_urlosRX);
faseAf_rad_urlosRX = angle(antena_urlosRX);
faseAf_deg_urlosRX = rad2deg(faseAf_rad_urlosRX);
AF1_RX = zeros(num_u,360);

for Rx_urlos = 1:num_u
    beta_urlosRX = 2*pi;
    d_urlosRX = 1/2;
    N1_urlosRX = 16;
    i = sqrt(-1);
    An_urlosRX = amplitudoAF_urlosRX(Rx_urlos,:);
    alpha_urlosRX                                  =
faseAf_rad_urlosRX(Rx_urlos,:);
    for theta_urlosRX = 1 : 360
        rad(theta_urlosRX)                         =
(theta_urlosRX*pi)/180;
        for n_RXurlos = 1 : N1_urlosRX
            AF1_RX(Rx_urlos,theta_urlosRX)         =
AF1_RX(Rx_urlos,theta_urlosRX)                     +
An_urlosRX*exp(i*(n_RXurlos-1)*((beta_urlosRX   *
d_urlosRX    *    sin(rad(theta_urlosRX))    )    +
alpha_urlosRX));
        end
        AF1_RX(Rx_urlos,theta_urlosRX)             =
abs(AF1_RX(Rx_urlos,theta_urlosRX));
end
end
total_radiasiRx = sum(AF1_RX,1);
radiasi_urlosRX                                    =
total_radiasiRx/max(total_radiasiRx);

%%% PLOT POLA RADIASI PADA OUTPUT TX
antena_irf_Tx_FD = sum(Vt_irf_FD,2);
amplitudoAF_irf_TX_FD = abs(antena_irf_Tx_FD);
faseAf_rad_irfTX_FD = angle(antena_irf_Tx_FD);
faseAf_deg_irfTX_FD                                =
rad2deg(faseAf_rad_irfTX_FD);
AF1_irfTX_FD = zeros(num_m,360);

for Tx_irf_FD = 1:num_m
```

```
    beta_irfTX_FD = 2*pi;
    d_irfTX_FD = 1/2;
    N1_irfTX_FD = 64 ;
    i = sqrt(-1);
    An_irfTX_FD                                    =
amplitudoAF_irf_TX_FD(Tx_irf_FD,:);
    alpha_irfTX_FD                                 =
faseAf_rad_irfTX_FD(Tx_irf_FD,:);
    for theta_irfTX_FD = 1 : 360
        rad(theta_irfTX_FD)                        =
(theta_irfTX_FD*pi)/180;
        for n_TXirf_FD = 1 : N1_irfTX_FD

AF1_irfTX_FD(Tx_irf_FD,theta_irfTX_FD)             =
AF1_irfTX_FD(Tx_irf_FD,theta_irfTX_FD)             +
An_irfTX_FD*exp(i*(n_TXirf_FD-1)*((beta_irfTX_FD
*  d_irfTX_FD  *  sin(rad(theta_irfTX_FD))  )  +
alpha_irfTX_FD));
        end
        AF1_irfTX_FD(Tx_irf_FD,theta_irfTX_FD)  =
abs(AF1_irfTX_FD(Tx_irf_FD,theta_irfTX_FD));
end
end
total_radiasiTx_urlos_FD = sum(AF1_irfTX_FD,1);
radiasi_irfTX_FD                                   =
total_radiasiTx_urlos_FD/max(total_radiasiTx_url
os_FD);

%%% PLOT POLA RADIASI PADA OUTPUT RX
input_irfRx_FD = (H_irf*Vt_irf_FD)'*wrf_irf_FD;
antena_irfRX_FD = (sum(input_irfRx_FD,2));
amplitudoAF_irfRX_FD = abs(antena_irfRX_FD);
faseAf_rad_irfRX_FD = angle(antena_irfRX_FD);
faseAf_deg_irfRX_FD                                =
rad2deg(faseAf_rad_irfRX_FD);
AF1_irfRX_FD = zeros(num_u,360);

for Rx_irf_FD = 1:Ns_FD
    beta_irfRX_FD = 2*pi;
    d_irfRX_FD = 1/2;
```

```
    N1_irfRX_FD = 16;
    i = sqrt(-1);
    An_irfRX_FD                                    =
amplitudoAF_irfRX_FD(Rx_irf_FD,:);
    alpha_irfRX_FD                                 =
faseAf_rad_irfRX_FD(Rx_irf_FD,:);
    for theta_irfRX_FD = 1 : 360
        rad(theta_irfRX_FD)                        =
(theta_irfRX_FD*pi)/180;
        for n_RXirf_FD = 1 : N1_irfRX_FD

AF1_irfRX_FD(Rx_irf_FD,theta_irfRX_FD)             =
AF1_irfRX_FD(Rx_irf_FD,theta_irfRX_FD)             +
An_irfRX_FD*exp(i*(n_RXirf_FD-1)*((beta_irfRX_FD
*  d_irfRX_FD  *  sin(rad(theta_irfRX_FD))  )  +
alpha_irfRX_FD));
        end
        AF1_irfRX_FD(Rx_irf_FD,theta_irfRX_FD)  =
abs(AF1_irfRX_FD(Rx_irf_FD,theta_irfRX_FD));
end
end
total_radiasiRx_urlos_FD = sum(AF1_irfRX_FD,1);
radiasi_irfRX_FD                                   =
total_radiasiRx_urlos_FD/max(total_radiasiRx_url
os_FD);

%%% PLOT POLA RADIASI PADA OUTPUT TX
antena_irf_Tx = sum(Vt_irfHB_us1,2);
amplitudoAF_irf_TX = abs(antena_irf_Tx);
faseAf_rad_irfTX = angle(antena_irf_Tx);
faseAf_deg_irfTX = rad2deg(faseAf_rad_irfTX);
AF1_irfTX_2us = zeros(num_m,360);

for Tx_irf = 1:num_m
    beta_irfTX = 2*pi;
    d_irfTX = 1/2;
    N1_irfTX = 64 ;
    i = sqrt(-1);
    An_irfTX = amplitudoAF_irf_TX(Tx_irf,:);
    alpha_irfTX = faseAf_rad_irfTX(Tx_irf,:);
```

```
    for theta_irfTX = 1 : 360
        rad(theta_irfTX) = (theta_irfTX*pi)/180;
        for n_TXirf = 1 : N1_irfTX
            AF1_irfTX_2us(Tx_irf,theta_irfTX)   =
AF1_irfTX_2us(Tx_irf,theta_irfTX)               +
An_irfTX*exp(i*(n_TXirf-1)*((beta_irfTX * d_irfTX
* sin(rad(theta_irfTX)) ) + alpha_irfTX));
        end
        AF1_irfTX_2us(Tx_irf,theta_irfTX)       =
abs(AF1_irfTX_2us(Tx_irf,theta_irfTX));
end
end
total_radiasiTX_irf2us = sum(AF1_irfTX_2us,1);
radiasi_irfTX_2us                               =
total_radiasiTX_irf2us/max(total_radiasiTX_irf2u
s);

%%% PLOT POLA RADIASI PADA OUTPUT RX user 1
input_irfRx_us1                                 =
(H_irf(1:num_u,:)*Vt_irfHB_us1)*wrf_irfHB_us1';
antena_irfRX_us1 = (sum(input_irfRx_us1,2));
amplitudoAF_irfRX_us1 = abs(antena_irfRX_us1);
faseAf_rad_irfRX_us1 = angle(antena_irfRX_us1);
faseAf_deg_irfRX_us1                            =
rad2deg(faseAf_rad_irfRX_us1);
AF1_irfRX1_us2 = zeros(num_u,360);

for Rx_irf_us1 = 1:num_u
    beta_irfRX_us1 = 2*pi;
    d_irfRX_us1 = 1/2;
    N1_irfRX_us1 = 16;
    i = sqrt(-1);
    An_irfRX_us1                                =
amplitudoAF_irfRX_us1(Rx_irf_us1,:);
    alpha_irfRX_us1                             =
faseAf_rad_irfRX_us1(Rx_irf_us1,:);
    for theta_irfRX_us1 = 1 : 360
        rad(theta_irfRX_us1)                    =
(theta_irfRX_us1*pi)/180;
        for n_RXirf_us1 = 1 : N1_irfRX_us1
```

```
AF1_irfRX1_us2(Rx_irf_us1,theta_irfRX_us1)      =
AF1_irfRX1_us2(Rx_irf_us1,theta_irfRX_us1)      +
An_irfRX_us1*exp(i*(n_RXirf_us1-
1)*((beta_irfRX_us1      *      d_irfRX_us1      *
sin(rad(theta_irfRX_us1)) ) + alpha_irfRX_us1));
        end

AF1_irfRX1_us2(Rx_irf_us1,theta_irfRX_us1)      =
abs(AF1_irfRX1_us2(Rx_irf_us1,theta_irfRX_us1));
end
end
total_radiasiRX1_irf2us = sum(AF1_irfRX1_us2,1);
radiasi_irfRX1_us2                              =
total_radiasiRX1_irf2us/max(total_radiasiRX1_irf
2us);

%%% PLOT POLA RADIASI PADA OUTPUT RX user 2
input_irfRx_us2                                 =
(H_irf((num_u+1):total_rx,:)*Vt_irfHB_us2)*wrf_i
rfHB_us2';
antena_irfRX_us2 = (sum(input_irfRx_us2,2));
amplitudoAF_irfRX_us2 = abs(antena_irfRX_us2);
faseAf_rad_irfRX_us2 = angle(antena_irfRX_us2);
faseAf_deg_irfRX_us2                            =
rad2deg(faseAf_rad_irfRX_us2);
AF1_irfRX_us2 = zeros(num_u,360);

for Rx_irf_us2 = 1:num_u
    beta_irfRX_us2 = 2*pi;
    d_irfRX_us2 = 1/2;
    N1_irfRX_us2 = 16;
    i = sqrt(-1);
    An_irfRX_us2                                =
amplitudoAF_irfRX_us2(Rx_irf_us2,:);
    alpha_irfRX_us2                             =
faseAf_rad_irfRX_us2(Rx_irf_us2,:);
    for theta_irfRX_us2 = 1 : 360
        rad(theta_irfRX_us2)                    =
(theta_irfRX_us2*pi)/180;
```

```
        for n_RXirf_us2 = 1 : N1_irfRX_us2

AF1_irfRX_us2(Rx_irf_us2,theta_irfRX_us2)       =
AF1_irfRX_us2(Rx_irf_us2,theta_irfRX_us2)       +
An_irfRX_us2*exp(i*(n_RXirf_us2-
1)*((beta_irfRX_us2      *      d_irfRX_us2      *
sin(rad(theta_irfRX_us2)) ) + alpha_irfRX_us2));
        end

AF1_irfRX_us2(Rx_irf_us2,theta_irfRX_us2)       =
abs(AF1_irfRX_us2(Rx_irf_us2,theta_irfRX_us2));
end
end
total_radiasiRX2_irf2us = sum(AF1_irfRX_us2,1);
radiasi_irfRX2_us2                              =
total_radiasiRX2_irf2us/max(total_radiasiRX2_irf
2us);

%%% PLOT POLA RADIASI PADA OUTPUT TX
antena_irf_Tx_FD = sum(Vt_irfFD_us1,2);
amplitudoAF_irf_TX_FD = abs(antena_irf_Tx_FD);
faseAf_rad_irfTX_FD = angle(antena_irf_Tx_FD);
faseAf_deg_irfTX_FD                             =
rad2deg(faseAf_rad_irfTX_FD);
AF1_irfTX_2us_FD = zeros(num_m,360);

for Tx_irf_FD = 1:num_m
    beta_irfTX_FD = 2*pi;
    d_irfTX_FD = 1/2;
    N1_irfTX_FD = 64 ;
    i = sqrt(-1);
    An_irfTX_FD                                 =
amplitudoAF_irf_TX_FD(Tx_irf_FD,:);
    alpha_irfTX_FD                              =
faseAf_rad_irfTX_FD(Tx_irf_FD,:);
    for theta_irfTX_FD = 1 : 360
        rad(theta_irfTX_FD)                     =
(theta_irfTX_FD*pi)/180;
        for n_TXirf_FD = 1 : N1_irfTX_FD
```

```
AF1_irfTX_2us_FD(Tx_irf_FD,theta_irfTX_FD)      =
AF1_irfTX_2us_FD(Tx_irf_FD,theta_irfTX_FD)      +
An_irfTX_FD*exp(i*(n_TXirf_FD-1)*((beta_irfTX_FD
*  d_irfTX_FD  *  sin(rad(theta_irfTX_FD))  )  +
alpha_irfTX_FD));
        end

AF1_irfTX_2us_FD(Tx_irf_FD,theta_irfTX_FD)      =
abs(AF1_irfTX_2us_FD(Tx_irf_FD,theta_irfTX_FD));
end
end
total_radiasiTX_irf2us_FD                       =
sum(AF1_irfTX_2us_FD,1);
radiasi_irfTX_2us_FD                            =
total_radiasiTX_irf2us_FD/max(total_radiasiTX_ir
f2us_FD);

%%% PLOT POLA RADIASI PADA OUTPUT RX user 1
input_irfRx_us1_FD                              =
(H_irf(1:num_u,:)*Vt_irfFD_us1)'*wrf_irfFD_us1;
antena_irfRX_us1_FD                             =
(sum(input_irfRx_us1_FD,2));
amplitudoAF_irfRX_us1_FD                        =
abs(antena_irfRX_us1_FD);
faseAf_rad_irfRX_us1_FD                         =
angle(antena_irfRX_us1_FD);
faseAf_deg_irfRX_us1_FD                         =
rad2deg(faseAf_rad_irfRX_us1_FD);
AF1_irfRX1_us2_FD = zeros(num_u,360);

for Rx_irf_us1_FD = 1:NRF_FD
    beta_irfRX_us1_FD = 2*pi;
    d_irfRX_us1_FD = 1/2;
    N1_irfRX_us1_FD = 16;
    i = sqrt(-1);
    An_irfRX_us1_FD                             =
amplitudoAF_irfRX_us1_FD(Rx_irf_us1_FD,:);
    alpha_irfRX_us1_FD                          =
faseAf_rad_irfRX_us1_FD(Rx_irf_us1_FD,:);
    for theta_irfRX_us1_FD = 1 : 360
```

```
        rad(theta_irfRX_us1_FD)                 =
(theta_irfRX_us1_FD*pi)/180;
        for n_RXirf_us1_FD = 1 : N1_irfRX_us1_FD

AF1_irfRX1_us2_FD(Rx_irf_us1_FD,theta_irfRX_us1_
FD)                                             =
AF1_irfRX1_us2_FD(Rx_irf_us1_FD,theta_irfRX_us1_
FD)    +    An_irfRX_us1_FD*exp(i*(n_RXirf_us1_FD-
1)*((beta_irfRX_us1_FD    *    d_irfRX_us1_FD    *
sin(rad(theta_irfRX_us1_FD))          )          +
alpha_irfRX_us1_FD));
        end

AF1_irfRX1_us2_FD(Rx_irf_us1_FD,theta_irfRX_us1_
FD)                                             =
abs(AF1_irfRX1_us2_FD(Rx_irf_us1_FD,theta_irfRX_
us1_FD));
end
end
total_radiasiRX1_irf2us_FD                      =
sum(AF1_irfRX1_us2_FD,1);
radiasi_irfRX1_us2_FD                           =
total_radiasiRX1_irf2us_FD/max(total_radiasiRX1_
irf2us_FD);

%%% PLOT POLA RADIASI PADA OUTPUT RX user 2
input_irfRx_us2_FD                              =
(H_irf((num_u+1):total_rx,:)*Vt_irfFD_us2)'*wrf_
irfFD_us2;
antena_irfRX_us2_FD                             =
(sum(input_irfRx_us2_FD,2));
amplitudoAF_irfRX_us2_FD                        =
abs(antena_irfRX_us2_FD);
faseAf_rad_irfRX_us2_FD                         =
angle(antena_irfRX_us2_FD);
faseAf_deg_irfRX_us2_FD                         =
rad2deg(faseAf_rad_irfRX_us2_FD);
AF1_irfRX_us2_FD = zeros(num_u,360);

for Rx_irf_us2_FD = 1:NRF_FD
```

```
    beta_irfRX_us2_FD = 2*pi;
    d_irfRX_us2_FD = 1/2;
    N1_irfRX_us2_FD = 16;
    i = sqrt(-1);
    An_irfRX_us2_FD                                =
amplitudoAF_irfRX_us2_FD(Rx_irf_us2_FD,:);
    alpha_irfRX_us2_FD                             =
faseAf_rad_irfRX_us2_FD(Rx_irf_us2_FD,:);
    for theta_irfRX_us2_FD = 1 : 360
        rad(theta_irfRX_us2_FD)                    =
(theta_irfRX_us2_FD*pi)/180;
        for n_RXirf_us2_FD = 1 : N1_irfRX_us2_FD

AF1_irfRX_us2_FD(Rx_irf_us2_FD,theta_irfRX_us2_F
D)                                                 =
AF1_irfRX_us2_FD(Rx_irf_us2_FD,theta_irfRX_us2_F
D)    +    An_irfRX_us2_FD*exp(i*(n_RXirf_us2_FD-
1)*((beta_irfRX_us2_FD    *    d_irfRX_us2_FD    *
sin(rad(theta_irfRX_us2_FD))          )          +
alpha_irfRX_us2_FD));
        end

AF1_irfRX_us2_FD(Rx_irf_us2_FD,theta_irfRX_us2_F
D)                                                 =
abs(AF1_irfRX_us2_FD(Rx_irf_us2_FD,theta_irfRX_u
s2_FD));
end
end
total_radiasiRX2_irf2us_FD                         =
sum(AF1_irfRX_us2_FD,1);
radiasi_irfRX2_us2_FD                              =
total_radiasiRX2_irf2us_FD/max(total_radiasiRX2_
irf2us_FD);

%%% PLOT POLA RADIASI PADA OUTPUT TX
antena_irf_Tx = sum(Vt_irfHB_us1,2);
amplitudoAF_irf_TX = abs(antena_irf_Tx);
faseAf_rad_irfTX = angle(antena_irf_Tx);
faseAf_deg_irfTX = rad2deg(faseAf_rad_irfTX);
AF1_irfTX_2us = zeros(num_m,360);
```

```
for Tx_irf = 1:num_m
    beta_irfTX = 2*pi;
    d_irfTX = 1/2;
    N1_irfTX = 64 ;
    i = sqrt(-1);
    An_irfTX = amplitudoAF_irf_TX(Tx_irf,:);
    alpha_irfTX = faseAf_rad_irfTX(Tx_irf,:);
    for theta_irfTX = 1 : 360
        rad(theta_irfTX) = (theta_irfTX*pi)/180;
        for n_TXirf = 1 : N1_irfTX
            AF1_irfTX_2us(Tx_irf,theta_irfTX)   =
AF1_irfTX_2us(Tx_irf,theta_irfTX)               +
An_irfTX*exp(i*(n_TXirf-1)*((beta_irfTX * d_irfTX
* sin(rad(theta_irfTX)) ) + alpha_irfTX));
        end
        AF1_irfTX_2us(Tx_irf,theta_irfTX)       =
abs(AF1_irfTX_2us(Tx_irf,theta_irfTX));
end
end
total_radiasiTX_irf2us = sum(AF1_irfTX_2us,1);
radiasi_irfTX_2us                               =
total_radiasiTX_irf2us/max(total_radiasiTX_irf2u
s);

%%% PLOT POLA RADIASI PADA OUTPUT RX user 1
input_irfRx_us1                                 =
(H_irf(1:num_u,:)*Vt_irfHB_us1)*wrf_irfHB_us1';
antena_irfRX_us1 = (sum(input_irfRx_us1,2));
amplitudoAF_irfRX_us1 = abs(antena_irfRX_us1);
faseAf_rad_irfRX_us1 = angle(antena_irfRX_us1);
faseAf_deg_irfRX_us1                            =
rad2deg(faseAf_rad_irfRX_us1);
AF1_irfRX1_us2 = zeros(num_u,360);

for Rx_irf_us1 = 1:num_u
    beta_irfRX_us1 = 2*pi;
    d_irfRX_us1 = 1/2;
    N1_irfRX_us1 = 16;
    i = sqrt(-1);
```

```
    An_irfRX_us1                                    =
amplitudoAF_irfRX_us1(Rx_irf_us1,:);
    alpha_irfRX_us1                                 =
faseAf_rad_irfRX_us1(Rx_irf_us1,:);
    for theta_irfRX_us1 = 1 : 360
        rad(theta_irfRX_us1)                        =
(theta_irfRX_us1*pi)/180;
        for n_RXirf_us1 = 1 : N1_irfRX_us1

AF1_irfRX1_us2(Rx_irf_us1,theta_irfRX_us1)      =
AF1_irfRX1_us2(Rx_irf_us1,theta_irfRX_us1)      +
An_irfRX_us1*exp(i*(n_RXirf_us1-
1)*((beta_irfRX_us1      *      d_irfRX_us1      *
sin(rad(theta_irfRX_us1)) ) + alpha_irfRX_us1));
        end

AF1_irfRX1_us2(Rx_irf_us1,theta_irfRX_us1)      =
abs(AF1_irfRX1_us2(Rx_irf_us1,theta_irfRX_us1));
end
end
total_radiasiRX1_irf2us = sum(AF1_irfRX1_us2,1);
radiasi_irfRX1_us2                              =
total_radiasiRX1_irf2us/max(total_radiasiRX1_irf
2us);

%%% PLOT POLA RADIASI PADA OUTPUT RX user 2
input_irfRx_us2                                 =
(H_irf((num_u+1):total_rx,:)*Vt_irfHB_us2)*wrf_i
rfHB_us2';
antena_irfRX_us2 = (sum(input_irfRx_us2,2));
amplitudoAF_irfRX_us2 = abs(antena_irfRX_us2);
faseAf_rad_irfRX_us2 = angle(antena_irfRX_us2);
faseAf_deg_irfRX_us2                            =
rad2deg(faseAf_rad_irfRX_us2);
AF1_irfRX_us2 = zeros(num_u,360);

for Rx_irf_us2 = 1:num_u
    beta_irfRX_us2 = 2*pi;
    d_irfRX_us2 = 1/2;
    N1_irfRX_us2 = 16;
```

```
    i = sqrt(-1);
    An_irfRX_us2                                    =
amplitudoAF_irfRX_us2(Rx_irf_us2,:);
    alpha_irfRX_us2                                 =
faseAf_rad_irfRX_us2(Rx_irf_us2,:);
    for theta_irfRX_us2 = 1 : 360
        rad(theta_irfRX_us2)                        =
(theta_irfRX_us2*pi)/180;
        for n_RXirf_us2 = 1 : N1_irfRX_us2

AF1_irfRX_us2(Rx_irf_us2,theta_irfRX_us2)       =
AF1_irfRX_us2(Rx_irf_us2,theta_irfRX_us2)       +
An_irfRX_us2*exp(i*(n_RXirf_us2-
1)*((beta_irfRX_us2      *      d_irfRX_us2      *
sin(rad(theta_irfRX_us2)) ) + alpha_irfRX_us2));
        end

AF1_irfRX_us2(Rx_irf_us2,theta_irfRX_us2)       =
abs(AF1_irfRX_us2(Rx_irf_us2,theta_irfRX_us2));
end
end
total_radiasiRX2_irf2us = sum(AF1_irfRX_us2,1);
radiasi_irfRX2_us2                              =
total_radiasiRX2_irf2us/max(total_radiasiRX2_irf
2us);

%%%% plot 100 sampel awal
figure(1);
stem(transpose(s_TX(1,1:100)));
title('BPSK Bit Data Sample');
xlabel('Jumlah Bit Data');
ylabel('Amplitudo');

%%%% plot 100 sampel akhir
figure(2);
stem(transpose(s_TX(1,1:100)));
title('BPSK Bit Data Sample');
xlabel('Jumlah Bit Data');
ylabel('Amplitudo');
```

```
%%% PLOT GRAFIK SPECTRAL EFFICIENCY VS SNR KANAL
IRF
figure(3);
plot(SNR_dB, mean(SE_IRF_FDB,2), 'bo-'); hold on;
plot(SNR_dB, mean(SE_IRF_HB,2), 'rx-'); hold on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('Independent Rayleigh Fading');
legend('Full     Digital     Beamforming','Hybrid
Beamforming');

%%% PLOT GRAFIK BER VS SNR KANAL IRF
figure(4);
semilogy(SNR_dB,mean(simBER_IRF_FDB,1),'bo-');
hold on;
semilogy(SNR_dB,mean(simBER_IRF_HB,1),'rx-');
hold on;
axis([-30 30 10^-4 1])
grid on
legend('Full-Digital         Beamforming','Hybrid
Beamforming');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal IRF pada sistem Point-to-Point
MIMO');

%%% PLOT GRAFIK SPECTRAL EFFICIENCY VS SNR KANAL
URLOS
figure(5);
plot(SNR_dB,  mean(SE_URLOS_FDB,2),  'bo-');  hold
on;
plot(SNR_dB,  mean(SE_URLOS_HB,2),  'rx-');  hold
on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('Uniformly Random Line-of-Sight');
legend('Full     Digital     Beamforming','Hybrid
Beamforming');
```

```
%%% PLOT GRAFIK BER VS SNR KANAL URLOS
figure(6)
semilogy(SNR_dB,mean(simBER_URLOS_FDB,1),'bo-');
hold on;
semilogy(SNR_dB,mean(simBER_URLOS_HB,1),'rx-');
hold on;
axis([-30 30 10^-4 1])
grid on
legend('Full-Digital        Beamforming','Hybrid
Beamforming');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal URLoS pada sistem Point-to-Point
MIMO');

figure(7);
plot(SNR_dB,  mean(SE_IRF_TX64,2),  'b.-');  hold
on;
plot(SNR_dB,  mean(SE_IRF_TX32,2),  'rx-');  hold
on;
plot(SNR_dB,  mean(SE_IRF_TX16,2),  'go-');  hold
on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('Independent Rayleigh Fading');
legend('Tx=64','Tx=32','Tx=16');

%%% PLOT GRAFIK BER VS SNR
figure(8)
semilogy(SNR_dB,mean(simBER_IRF_TX64,1),'b.-');
hold on;
semilogy(SNR_dB,mean(simBER_IRF_TX32,1),'rx-');
hold on;
semilogy(SNR_dB,mean(simBER_IRF_TX16,1),'go-');
hold on;
axis([-30 30 10^-4 1])
grid on
legend('Tx=64','Tx=32','Tx=16');
```

```
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal IRF Point-to-Point MIMO');

figure(9);
plot(SNR_dB, mean(SE_urlos_TX64,2), 'b.-'); hold
on;
plot(SNR_dB, mean(SE_urlos_TX32,2), 'rx-'); hold
on;
plot(SNR_dB, mean(SE_urlos_TX16,2), 'go-'); hold
on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('Uniformly Random Line-of-Sight');
legend('Tx=64','Tx=32','Tx=16');

%%% PLOT GRAFIK BER VS SNR
figure(10)
semilogy(SNR_dB,mean(simBER_urlos_TX64,1),'b.-
'); hold on;
semilogy(SNR_dB,mean(simBER_urlos_TX32,1),'rx-
'); hold on;
semilogy(SNR_dB,mean(simBER_urlos_TX16,1),'go-
'); hold on;
axis([-30 30 10^-4 1])
grid on
legend('Tx=64','Tx=32','Tx=16');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal URLoS Point-to-Point MIMO');

figure(11);
plot(SNR_dB, mean(SE_IRF_RX64,2), 'b.-'); hold
on;
plot(SNR_dB, mean(SE_IRF_RX32,2), 'rx-'); hold
on;
plot(SNR_dB, mean(SE_IRF_RX16,2), 'go-'); hold
on;
grid on;
```

```
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('Independent Rayleigh Fading');
legend('RX=16','RX=12','RX=8');

%%% PLOT GRAFIK BER VS SNR
figure(12)
semilogy(SNR_dB,mean(simBER_irf_RX64,1),'b.-');
hold on;
semilogy(SNR_dB,mean(simBER_irf_RX32,1),'rx-');
hold on;
semilogy(SNR_dB,mean(simBER_irf_RX16,1),'go-');
hold on;
axis([-30 30 10^-4 1])
grid on
legend('RX=16','RX=12','RX=8');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal IRF Point-to-Point MIMO');

figure(13);
plot(SNR_dB, mean(SE_urlos_RX64,2), 'b.-'); hold
on;
plot(SNR_dB, mean(SE_urlos_RX32,2), 'rx-'); hold
on;
plot(SNR_dB, mean(SE_urlos_RX16,2), 'go-'); hold
on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('Uniformly Random Line-of-Sight');
legend('RX=16','RX=12','RX=8');

%%% PLOT GRAFIK BER VS SNR
figure(14)
semilogy(SNR_dB,mean(simBER_urlos_RX64,1),'b.-
'); hold on;
semilogy(SNR_dB,mean(simBER_urlos_RX32,1),'rx-
'); hold on;
semilogy(SNR_dB,mean(simBER_urlos_RX16,1),'go-
```

```
'); hold on;
axis([-30 30 10^-4 1])
grid on
legend('RX=16','RX=12','RX=8');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal URLoS Point-to-Point MIMO');

%%% PLOT GRAFIK SPECTRAL EFFICIENCY VS SNR
figure(15);
plot(SNR_dB, mean(SE_IRF_FDB,2), 'b.-'); hold on;
plot(SNR_dB, mean(SE_IRF_HB,2), 'rx-'); hold on;
plot(SNR_dB, mean(SE_IRF_RF4,2), 'go-'); hold on;
plot(SNR_dB, mean(SE_IRF_RF2,2), 'md-'); hold on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('SE point-to-point MIMO kanal IRF dengan
variasi jumlah RF Chain');
legend('Full        Digital        Beamforming(RF
Chain=12)','Hybrid  Beamforming(RF  Chain=6)','RF
Chain=4','RF Chain=2');

%%% PLOT GRAFIK BER VS SNR
figure(16)
semilogy(SNR_dB,mean(simBER_IRF_FDB,1),'b.-');
hold on;
semilogy(SNR_dB,mean(simBER_IRF_HB,1),'rx-');
hold on;
semilogy(SNR_dB,mean(simBER_irf_RF4,1),'go-');
hold on;
semilogy(SNR_dB,mean(simBER_irf_RF2,1),'md-');
hold on;
axis([-30 30 10^-4 1])
grid on
legend('Full        Digital        Beamforming(RF
Chain=12)','Hybrid  Beamforming(RF  Chain=6)','RF
Chain=4','RF Chain=2');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
```

```
title('BER Kanal IRF pada sistem Point-to-Point
MIMO');

%%% PLOT GRAFIK SPECTRAL EFFICIENCY VS SNR
figure(17);
plot(SNR_dB, mean(SE_URLOS_FDB,2), 'b.-'); hold
on;
plot(SNR_dB, mean(SE_URLOS_HB,2), 'rx-'); hold
on;
plot(SNR_dB, mean(SE_URLOS_RF4,2), 'go-'); hold
on;
plot(SNR_dB, mean(SE_URLOS_RF2,2), 'md-'); hold
on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('SE point-to-point MIMO kanal URLOS dengan
variasi jumlah RF Chain');
legend('Full Digital Beamforming(RF
Chain=12)','Hybrid Beamforming(RF Chain=6)','RF
Chain=4','RF Chain=2');

%%% PLOT GRAFIK BER VS SNR
figure(18)
semilogy(SNR_dB,mean(simBER_URLOS_FDB,1),'b.-');
hold on;
semilogy(SNR_dB,mean(simBER_URLOS_HB,1),'rx-');
hold on;
semilogy(SNR_dB,mean(simBER_urlos_RF4,1),'go-');
hold on;
semilogy(SNR_dB,mean(simBER_urlos_RF2,1),'md-');
hold on;
axis([-30 30 10^-4 1])
grid on
legend('Full Digital Beamforming(RF
Chain=12)','Hybrid Beamforming(RF Chain=6)','RF
Chain=4','RF Chain=2');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal URLOS pada sistem Point-to-Point
```

```
MIMO');

%%% PLOT GRAFIK SPECTRAL EFFICIENCY VS SNR
figure(19);
plot(SNR_dB, mean(SE_IRF_FDB2us,2), 'b.-'); hold
on;
plot(SNR_dB, mean(SE_IRF_HB2us,2), 'rx-'); hold
on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
title('SE HB dan FDB pada kanal IRF untuk 2
User');
legend('Full Digital Beamforming','Hybrid
Beamforming');

%%% PLOT GRAFIK BER VS SNR
figure(20)
semilogy(SNR_dB,mean(simBER_irf_FDBus2,1),'b.-
'); hold on;
semilogy(SNR_dB,mean(simBER_irf_HBus2,1),'rx-');
hold on;
axis([-30 30 10^-4 1])
grid on
legend('Full Digital Beamforming','Hybrid
Beamforming');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal IRF pada sistem Point-to-Point
LSA-MIMO untuk 2 User');

%%% PLOT GRAFIK SPECTRAL EFFICIENCY VS SNR
figure(21);
plot(SNR_dB, mean(SE_urlos_FDBus2,2), 'b.-');
hold on;
plot(SNR_dB, mean(SE_urlos_HBus2,2), 'rx-'); hold
on;
grid on;
xlabel('SNR (dB)');
ylabel('Spectral Efficiency (bits/s/Hz)');
```

```
title('Uniformly Random Line-of-Sight');
legend('Full     Digital     Beamforming','Hybrid
Beamforming');

%%% PLOT GRAFIK BER VS SNR
figure(22)
semilogy(SNR_dB,mean(simBER_FDBus2,1),'b.-');
hold on;
semilogy(SNR_dB,mean(simBER_HBus2,1),'rx-');
hold on;
axis([-30 30 10^-4 1])
grid on
legend('Full     Digital     Beamforming','Hybrid
Beamforming');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
title('BER Kanal urlos pada sistem Point-to-Point
MIMO');

%%% PLOT POLA RADIASI PADA INPUT TX
figure(23);
ax = polaraxes;
polarplot(ax,radiasi_irfTX_HB);
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('Pola Radiasi TX sistem berbasis HB pada
kanal IRF');
thetalim([-90 90]);

%%% PLOT POLA RADIASI PADA INPUT RX
figure(24);
ax = polaraxes;
polarplot(ax,flip(radiasi_RX_HB,2));
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('Pola Radiasi RX sistem berbasis HB pada
kanal IRF');
thetalim([-90 90]);

%%% PLOT POLA RADIASI PADA INPUT TX
```

```
figure(25);
ax = polaraxes;
polarplot(ax,radiasi_irfTX_FD);
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('Pola Radiasi TX sistem berbasis FDB pada
kanal IRF');
thetalim([-90 90]);

%%% PLOT POLA RADIASI PADA INPUT TX
figure(26);
ax = polaraxes;
polarplot(ax,flip(radiasi_irfRX_FD,2));
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('Pola Radiasi RX sistem berbasis FDB pada
kanal IRF ');
thetalim([-90 90]);

%%% PLOT POLA RADIASI PADA INPUT TX
figure(27);
ax = polaraxes;
polarplot(ax,radiasi_urlosTX);
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('Pola Radiasi TX sistem berbasis FDB pada
kanal URLoS');
thetalim([-90 90]);

%%% PLOT POLA RADIASI PADA INPUT RX
figure(28);
ax = polaraxes;
polar(ax,radiasi_RX);
ax.ThetaDir = 'clockwise';
ax.ThetaZeroLocation = 'top';
title('Pola Radiasi TX sistem berbasis FDB pada
kanal URLoS');
thetalim([-90 90]);
```

BIOGRAFI PENULIS

Penulis buku Tugas Akhir dengan judul "**Analisis Kinerja Sistem *Point-to-Point Large-Scale Array* MIMO berbasis *Hybrid Beamforming* pada kanal *Rayleigh* dan UR-LoS**" bernama lengkap Putu Agus Kartika Adhi Pratama. Penulis lahir di Bogor, 24 November 1998. Penulis menempuh Pendidikan dasar di Sekolah Dian Harapan, kemudian melanjutkan di SMPN 1 Cikarang Utara, lalu pada pertengahan kelas 8 pindah ke Bontang, Kalimantan Timut dan melanjutkan sekolah di SMP YPVDP Bontang dan SMA YPVDP Bontang sampai tahun 2016. Setelah lulus SMA, penulis melanjutkan Pendidikan tingkat tinggi di Institut Teknologi Sepuluh Nopember, Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas. Selama masa perkuliahan, penulis aktif di beberapa organisasi kampus seperti TPKH-ITS, SPE ITS SC, dan BEM FTE ITS. Penulis juga aktif mengikuti beberapa lomba, seperti lomba nasional *Oil Rig Design Competition* IPFEST 2019 yang berhasil mencapai babak final, serta lomba internasional IEEE ComSoc *Student Competition* 2019 mewakili tim ITS yang berhasil masuk 15 besar dan meraih piagam "*Honorary Mention*".