

PROYECTO 3: SIMULACIÓN SISTEMAS TDMA CON SCHEDULING ADAPTATIVO

ABDÍAS ARAGÓN RAFOLS, SERGIO RAMÍREZ RODAS Y GONZALO PACHECO AGREDANO

Distribución de usuarios:

10 usuarios:

- Situados a la misma distancia de 150 m de la estación base
- Aleatoriamente y uniformemente repartidos en un área cuadrada de 800x800 m²

Datos del sistema:

Canal Rayleigh plano en frecuencia, selectivo en el tiempo, con ancho de banda de 10 MHz y una frecuencia de trabajo de 1.8 GHz. El patrón de movilidad es Jakes y la frecuencia Doppler máxima se detalla más adelante.

Los símbolos se agrupan en bloques (tramas) de tamaño 32 y se consideran los dos algoritmos de scheduling adaptativos siguientes:

- 1) Un algoritmo que asigna el acceso al usuario que tiene una mejor SNR en el primer instante de la trama: $Ui=SNR_i$, donde SNR_i representa la SNR instantánea en el primer instante de la trama.
- 2) Un algoritmo que asigna el acceso al usuario que presenta un mayor valor de utilidad, donde la utilidad del usuario i se define como: $Ui=SNR_i + c/(20-d)$. En la fórmula anterior, SNR_i representa la SNR instantánea en el primer instante de la trama, c es una constante que es igual a 5 veces la SNR media del canal y d es el número de tramas que transcurrieron desde que el usuario transmitió por última vez. Si $d>19$, el valor de Ui es $2c$. Si dos usuarios alcanzan el mismo valor de Ui , se escoge de forma aleatoria el que accede al canal.

Estime a través de simulaciones la PDF de:

- la SNR del canal de cada uno de los usuarios.
- el número de tramas que tiene que esperar un usuario hasta que dicho usuario vuelve a transmitir (retardo instantáneo).

Considere canales con 3 frecuencias Doppler diferentes:

- i) 5 Hz.
- ii) 50 Hz
- iii) 120 Hz.

Simulación:

- a) Para la primera distribución de usuarios obtenga:
 - i) La PDF correspondientes a la SNR media para todos los usuarios.
 - ii) Las 3 PDFs correspondientes al número de tramas de espera para todos los usuarios en cada frecuencia Doppler.

- iii) Evalúe 106 realizaciones del canal con los efectos de pequeña escala. Suponga que todos los usuarios utilizan una modulación 16QAM sin codificar, utilice las PDFs para calcular: el retardo medio y la BER media.
- b) Repita el apartado a) considerando la distribución aleatoria de usuarios. Realice un Monte Carlo para 100 distribuciones diferentes de usuarios y obtenga las CDF del retardo medio, la BER media y la tasa transmitida a cada usuario en ambas distribuciones de usuarios.
- c) Considere ahora que el usuario que transmite en cada instante puede utilizar la modulación que maximice la eficiencia espectral garantizando una BER<10⁻³ entre la 16QAM y la 64QAM. Obtenga la CDF de la nueva tasa transmitida a cada usuario y compárela con el resultado obtenido en el apartado b).

DATOS COMUNES PARA TODOS LOS PROYECTOS

Potencia de transmisión = 20 W

Densidad espectral de potencia de ruido en recepción = 10-20 W/Hz

Modelo de Path Loss: $32.4 + 20 \cdot \log_{10}(f\text{GHz}) + 37.6 \log_{10}(dm)$

Desviación típica log-normal del Shadowing: 1.5 dB

Aproximamos la BER instantánea mediante la fórmula: $\text{BER}(\text{SNR}) = 0.2 \exp(-\text{SNR}/(M-1))$

FUNCIONES REALIZADAS PARA EL PROYECTO

DISTRIBUCIÓN DE LOS USUARIOS

1.Calc_Pos_MS

Disponemos de dos tipos de distribución de usuarios:

La primera situados todos los usuarios a una distancia de 150 m de la estación base y otra donde los usuarios están repartidos de forma aleatoria y uniforme en un área cuadrada de 800x800 m².

Hemos creado un función llamada "Calc_Pos_MS" que mediante el parámetro "option" permite elegir entre las dos distribuciones.

```
function Pos_MS = Calc_Pos_MS(option, K, lado_x2, lado_y2, lado_x, lado_y)
% Calc_Pos_MS - Calcula la posición de los móviles (MS) según la opción.
% option = 1: Distribución uniforme en un área rectangular (lado_x x lado_y)
% option = 2: Distribución en círculo de radio lado_x2/lado_y2 respecto a
la estación base

Pos_MS = [];% Inicializar

if option == 1
    % Opción 1: usuarios uniformemente distribuidos en un área rectangular
    for k = 1:K
        Pos_MS(k) = lado_x* (rand - 0.5) + 1i * lado_y * (rand - 0.5);
    end
end
```

```

elseif option == 2
    % Opción 2: usuarios sobre un círculo (misma distancia al centro)
    Angle_MS = 2 * pi * rand(1, K);
    Pos_MS = lado_x2 * cos(Angle_MS) + 1i * lado_y2 * sin(Angle_MS);

else
    error('La opción elegida no es correcta. Elige entre 1 o 2.');
end

end

```

Esta función genera las posiciones de los móviles (MS) en una simulación de red TDMA, según las dos posibles distribuciones como he dicho anteriormente. Option, define el tipo de distribución espacial de los usuarios; K, número total de usuarios móviles (MS); lado_x y lado_y para escoger las dimensiones del área rectangular si option == 1; y lado_x2 y lado_y2 para los radios para generar un círculo si option == 2. El objetivo de la función es calcular y devolver un vector complejo Pos_MS que represente las posiciones de los móviles en el plano, usando números complejos.

La función sigue un comportamiento u otro según el valor de "option". La opción 1, es para seleccionar la distribución aleatoria uniforme en área rectangular. Para cada usuario k, se genera una posición aleatoria dentro de un rectángulo centrado en el origen. (rand - 0.5) genera un número entre -0.5 y 0.5. Al multiplicar por lado_x y lado_y se escala a las dimensiones reales del área. Se guarda como un número complejo: $(x + i*y)$.

La opción 2 es la de la distribución circular. Se generan K ángulos aleatorios uniformemente distribuidos entre 0 y 2π . Luego se calcula la posición de cada usuario como un punto en el círculo de radio $\text{lado_x2}/\text{lado_y2}$. El resultado es un conjunto de usuarios equidistantes al centro, pero en direcciones aleatorias.

En conclusión, el resultado es un vector Pos_MS de longitud K, donde cada elemento es un número complejo con la posición del usuario en el plano XY.

DATOS DEL SISTEMA

2. Apply_Rayleigh_Channel

Tenemos un canal Rayleigh, plano en frecuencia y selectivo en tiempo, con ancho de banda 10 MHz, y una frecuencia de trabajo de 1.8 GHz. Se utiliza un patrón de movilidad Jakes y la frecuencia Doppler máxima va a ir variando para distintos casos.

Para ello, generamos el canal con la siguiente función, "Apply_Rayleigh_Channel":

```

function SNR_instantanea = Apply_Rayleigh_Channel(SNR_dB, fd, num_muestras,
ts)
% Apply_Rayleigh_Channel – Aplica el canal Rayleigh y devuelve la SNR
instantánea.
% SNR_dB          : vector de SNR medias en dB (Kx1)
% fd              : frecuencia Doppler en Hz
% num_muestras   : número total de muestras a simular
% ts              : período de muestreo (s)
%

```

```

% Salida:
%   SNR_instantanea : matriz K×num_muestras con la SNR instantánea en dB
%                       para cada usuario en cada una de las num_muestras.

K = length(SNR_dB);
SNR_instantanea = zeros(K, num_muestras);

for n = 1:K
    % 1) SNR media en unidades naturales
    snr_nat = 10^(SNR_dB(n) / 10);

    % 2) Creamos el canal Rayleigh
    ch = comm.RayleighChannel( ...
        'SampleRate', 1/ts, ...
        'MaximumDopplerShift', fd, ...
        'PathGainsOutputPort', true);

    % 3) Generamos num_muestras ganancias complejas H
    [~, H] = ch( ones(num_muestras, 1) ); % H es num_muestras×1

    % 4) Escalamos H por sqrt(snr_linear): h(n,:) es 1×num_muestras
    h(n, :) = (sqrt(snr_nat) * H.').';

    % 5) Calculamos SNR instantánea en dB = 10·log10(|h(n,j)|^2)
    SNR_instantanea(n, :) = 10 * log10( abs(h(n, :)).^2 );
end
end

```

La función `Apply_Rayleigh_Channel` simula el efecto del desvanecimiento rápido (fading) causado por movilidad (modelo de Jakes) sobre una SNR media para obtener una SNR instantánea por símbolo y esto lo hacemos para cada usuario. Esto es clave para un canal Rayleigh selectivo en el tiempo. Utiliza un canal Rayleigh con variación temporal (Doppler) para modelar la movilidad de los usuarios.

3. Scheduler

Los símbolos se agrupan en bloques o tramas de tamaño 32, y vamos a tener dos algoritmos de scheduling adaptativos diferentes. Esto lo gestionamos con la función "Scheduler":

```

function [selected_user, selected_users, delay_vector, hist_SNR, retardos,
user_tx_count] = Scheduler(SNR_instantanea, delay_vector, SNR_avg,
algoritmo, Num_tramas)
% Scheduler – Elige el usuario a programar según el algoritmo elegido,
considerando SNR instantánea cada 32 símbolos.
% Inputs:
%   SNR_instantanea – Matriz de SNRs instantáneas [dB], de tamaño K x
Num_tramas
%   delay_vector     – Vector con los retrasos acumulados [tramas]
%   SNR_avg          – SNR promedio (solo para utilidad)

```

```

%   algoritmo      - 'maxSNR' o 'utilidad'
%   Num_tramas     - Número total de tramas
% Outputs:
%   selected_user   - Último usuario seleccionado
%   selected_users  - Vector con todos los usuarios seleccionados por
trama
%   delay_vector    - Vector actualizado de retrasos
%   hist_SNR        - Historial de SNRs para cada usuario
%   retardos        - Historial de retrasos para cada usuario
%   user_tx_count   - Conteo de transmisiones por usuario

c = 5 * SNR_avg; % Constante c, basada en la SNR media

%Inicialización de estructuras de salida:
K = size(SNR_instantanea, 1); % Número de usuarios
selected_users = zeros(1, Num_tramas); % Para guardar qué usuario
transmite en cada trama
hist_SNR = cell(1, K); % Para guardar SNRs cuando
transmite cada usuario
retardos = cell(1, K); % Para guardar retrasos cuando
transmite cada usuario
user_tx_count = zeros(1, K); % Contador de transmisiones por
usuario

%Bucle por cada trama:
for i = 1:Num_tramas
    SNR_block = SNR_instantanea(:, i); % SNR del instante actual (columna
i)

% Elección del usuario según algoritmo:
switch lower(algoritmo)
    % Caso 'maxsnr':
    case 'maxsnr'
        % Selección del usuario con el mayor SNR en el bloque
        [~, selected_user] = max(SNR_block);

    case 'utilidad'
        % Calcular la utilidad para cada usuario en función de su
retraso
        d = delay_vector; % Donde d es el número de tramas desde la
última transmisión del usuario.
        d(d == 0) = 1; % Evitar división por cero

        % Fórmula de utilidad
        utilidad = SNR_block + c ./ (20 - d);

        % Penalización para retrasos mayores a 19
        idx = d >= 20;
        if any(idx)
            utilidad(idx) = 2 * c;

```

```

    end

    % Selección del usuario con la mayor utilidad
    max_u = max(utilidad);
    candidatos = find(utilidad == max_u);
    if length(candidatos) > 1
        selected_user = candidatos(randi(length(candidatos)));
    else
        selected_user = candidatos;
    end

    otherwise
        error('Algoritmo no reconocido.');
    end

selected_user = min(max(1, selected_user), K); % Seguridad

% Registrar el usuario seleccionado para esta trama
selected_users(i) = selected_user;

% Actualizar estadísticas
hist_SNR{selected_user} = [hist_SNR{selected_user},
SNR_block(selected_user)];
retardos{selected_user} = [retardos{selected_user},
delay_vector(selected_user)];

% Incrementar el contador de transmisiones del usuario seleccionado
user_tx_count(selected_user) = user_tx_count(selected_user) + 1;

% Actualizar el vector de retrasos
delay_vector = delay_vector + 1;
delay_vector(selected_user) = 0;
end

end

```

Esta función gestiona la asignación de acceso al canal en cada trama, basándose en dos posibles algoritmos adaptativos:

- MaxSNR: elige al usuario con mejor SNR en el instante actual.
- Utilidad: prioriza al usuario con mayor utilidad según SNR y cuántas tramas lleva sin transmitir.

Se calcula la utilidad como: $U_i = SNR_i + c/(20 - d)$.

Devuelve, para las Num_tramas:

- Qué usuario transmitió en cada trama (selected_users)
- Qué retardo llevaba cada usuario al transmitir (retardos)
- Qué SNR tuvo cada usuario al transmitir (hist_SNR)

- Cuántas veces transmitió cada usuario (user_tx_count).

ESTIMACIÓN DE LA PDF

4. calcular_SNRs

Ahora hay que estimar la PDF a través de simulaciones; la SNR del canal de cada uno de los usuarios; y el número de tramas que tiene que esperar un usuario hasta que dicho usuario vuelve a transmitir (retardo instantáneo).

- SNR del canal de cada uno de los usuarios

```
function SNR_dB = calcular_SNRs(Pos_BS, Pos_MS, Ptx, B, f, sigma_sh, N0)
% Calcula la SNR para cada usuario considerando el path loss, shadowing, y
Rayleigh fading.
%
% Inputs:
%   Pos_BS      - Posición de la estación base
%   Pos_MS      - Posiciones de los usuarios (vector)
%   Ptx         - Potencia de transmisión
%   B           - Ancho de banda del canal
%   f           - Frecuencia en Hz
%   sigma_sh    - Desviación estándar para el shadowing log-normal
%   N0          - Densidad espectral de ruido (W/Hz)
%
% Output:
%   SNR_dB     - SNR en dB para cada usuario

K = length(Pos_MS);                      % Número de usuarios
SNR_dB = zeros(1, K);                     % Vector para almacenar la SNR en dB

% Calculamos la potencia total de ruido (W)
Noise = B * N0;

% Calculamos la distancia entre la estación base y los usuarios
dist_BS_MS = abs(Pos_MS - Pos_BS);

% Calculamos Path Loss en dB
PL = 32.4 + 20*log10(f / 1e9) + 37.6 * log10(dist_BS_MS);      % Path
loss en dB
pl = 10.^ (PL / 10);                         % Path
loss en unidades naturales

% Calculamos la potencia recibida media (en dB)
Prx_dB = 10*log10(Ptx) - PL;                  %
Potencia recibida media en dB

% Aplicamos shadowing log-normal
```

```

Prx_dB = Prx_dB + sigma_sh * randn(1, K); % Potencia recibida (dB) aplicando shadowing

% Calculamos la SNR (en dB) sin Rayleigh fading
SNR_dB = Prx_dB - 10 * log10(Noise); % SNR
en dB
end

```

La función "calcular_SNRs" calcula la SNR media en dB para cada usuario en base a su distancia a la estación base, aplicando pérdidas por propagación (path loss), shadowing log-normal, y potencia de transmisión. Tiene en cuenta el ancho de banda, la frecuencia y el ruido.

5. plotMeanDelay

Con la salida de scheduler obtenemos la variable retardos que contiene un array de los retardos de cada usuario. Y para representarlo implementamos la función "plotMeanDelay".

```

function plotMeanDelay(retardos)
% plotMeanDelay – Calcula y dibuja el retardo medio (en tramas) por usuario.
%
% Input:
%   retardos – cell array 1×K donde cada elemento es el vector de retardos
%               (“número de tramas esperadas”) registrados para ese usuario

K = numel(retardos);
meanDelay = zeros(1, K);
for u = 1:K
    if isempty(retardos{u})
        meanDelay(u) = NaN; % Usuario nunca seleccionado
    else
        meanDelay(u) = mean(retardos{u});
    end
end

figure;
bar(1:K, meanDelay);
xlabel('Usuario');
ylabel('Retardo medio [tramas]');
title('Retardo medio hasta volver a transmitir por usuario');
grid on;
end

```

La función "plotMeanDelay" calcula el retardo medio en tramas para cada usuario a partir del historial de transmisiones y lo representa en un gráfico de barras. Es útil para visualizar cuánto tarda en transmitir cada usuario bajo el algoritmo de scheduling.

6. Transmit_16QAM

La función "Transmit_16QAM" simula la transmisión y recepción de datos usando modulación 16QAM. Para cada trama, modula 4 bits ($\log_2(M)$) del usuario seleccionado, transmite por el canal y luego demodula para recuperar los bits recibidos.

```
function [brx, y] = Transmit_16QAM(btx, selected_users, h, M)
% Transmit_16QAM – Transmite información usando 16QAM basado en los
usuarios seleccionados.
%   btx           – Matriz de bits a transmitir (K x num_bits)
%   selected_users – Vector de usuarios seleccionados para cada trama
(longitud num_tramas)
%   h             – Canal Rayleigh (K x num_tramas)
%   M             – Orden de la modulación (16 para 16QAM)

K = size(btx, 1);
num_tramas = length(selected_users);

% Inicializar variables
brx = zeros(size(btx)); % Bits recibidos
y = zeros(1, num_tramas); % Señal transmitida
nb_v = zeros(K, 1); % Contador de bits transmitidos por usuario

% MODULACIÓN Y TRANSMISIÓN
for n = 1:num_tramas
    % Identificar el usuario que transmite
    m_opt = selected_users(n);

    % Modular los bits del usuario seleccionado
    nb = nb_v(m_opt);
    bits = btx(m_opt, (nb+1):(nb+4));
    x_sched = qammod(bits.', M, 'InputType', 'bit');

    % Transmitir a través del canal
    y(n) = h(m_opt, n) * x_sched;

    % Actualizar el contador de bits
    nb_v(m_opt) = nb_v(m_opt) + 4;
end

% DEMODULACIÓN
nb_v = zeros(K, 1); % Reiniciar el contador de bits
for n = 1:num_tramas
    % Identificar el usuario que transmitió
    m_sch = selected_users(n);

    % Demodular los bits recibidos
    nb = nb_v(m_sch);
    received_bits = qamdemod(y(n) ./ h(m_sch, n), M, 'OutputType', 'bit');
    brx(m_sch, (nb+1):(nb+4)) = received_bits.';

```

```

% Actualizar el contador de bits
nb_v(m_sch) = nb_v(m_sch) + 4;
end

end

```

7. Transmit_AdaptiveQAM

La función `Transmit_AdaptiveQAM` simula la transmisión y recepción de datos con modulación adaptativa por trama. A diferencia de `Transmit_16QAM`, esta función permite cambiar la modulación dinámicamente entre 16QAM y 64QAM según la SNR instantánea del usuario seleccionado, garantizando una BER inferior a 10^{-3}

```

function [brx, y, tasa_k] = Transmit_AdaptiveQAM(btx, selected_users, h,
M_vector, B)
% Transmite con modulación adaptativa y devuelve la tasa transmitida por
usuario
% Inputs:
%   btx: bits transmitidos por cada usuario (K x total_bits)
%   selected_users: vector con el usuario seleccionado en cada trama
%   h: canal Rayleigh (K x num_tramas)
%   M_vector: vector de orden de modulación (1 x num_tramas)
%   B: ancho de banda para calcular la tasa [Mb/s]
% Output:
%   brx: bits transmitidos por cada usuario (K x total_bits)
%   y: la señal recibida
%   tasa_k: tasa transmitida por usuario

K = size(btx, 1);
num_tramas = length(selected_users);

brx = zeros(size(btx));
y = zeros(1, num_tramas);
nb_v = zeros(K, 1); % bits transmitidos por usuario
bit_count = zeros(K, 1); % contador total de bits transmitidos

% MODULACIÓN Y TRANSMISIÓN
for n = 1:num_tramas
    m_opt = selected_users(n);
    M = M_vector(n);
    bits_per_sym = log2(M);

    nb = nb_v(m_opt);
    bits = btx(m_opt, (nb+1):(nb+bits_per_sym));
    x_sched = qammod(bits.', M, 'InputType', 'bit');

    y(n) = h(m_opt, n) * x_sched;
    nb_v(m_opt) = nb_v(m_opt) + bits_per_sym;
    bit_count(m_opt) = bit_count(m_opt) + bits_per_sym;
end

```

```

end

% DEMODULACIÓN
nb_v = zeros(K, 1);
for n = 1:num_tramas
    m_sch = selected_users(n);
    M = M_vector(n);
    bits_per_sym = log2(M);

    nb = nb_v(m_sch);
    received_bits = qamdemod(y(n) ./ h(m_sch, n), M, 'OutputType', 'bit');
    brx(m_sch, (nb+1):(nb+bits_per_sym)) = received_bits.';
    nb_v(m_sch) = nb_v(m_sch) + bits_per_sym;
end

% Cálculo de tasa transmitida (bitrate en Mb/s)
duracion_total = num_tramas * 32 / B; % duración en segundos
tasa_k = bit_count / duracion_total / 1e6; % en Mb/s

end

```

8. Calculate_BER

Para calcular la BER media por usuario utilizamos la función "Calculate_BER" que calcula la BER media por usuario a partir de sus SNR instantáneas, utilizando la fórmula exponencial aproximada $BER = 0.2 \cdot \exp(-SNR/(M - 1))$. Primero convierte las SNRs instantáneas pasadas como parámetro de cada usuario en cada trama a unidades naturales y luego calcula la BER instantánea, gracias a la fórmula mencionada anteriormente, y finalmente calcula la BER media de cada usuario.

```

function BER_avg = Calculate_BER(SNR_instantanea, M)
% Calculate_BER – Calcula la BER media utilizando la fórmula especificada.
%   SNR_instantanea – Matriz de SNRs instantáneas [dB], de tamaño K x
Num_tramas
%   M           – Orden de la modulación (e.g., 16 para 16-QAM, 64 para
64-QAM)

% Convertir SNR de dB a unidades naturales
SNR_nat = 10.^ (SNR_instantanea / 10);

% Calcular la BER instantánea usando la fórmula dada
BER_instantanea = 0.2 * exp(-SNR_nat / (M - 1));

% Calcular la BER media para cada usuario
BER_avg = mean(BER_instantanea, 2);

end

```

9. Calculate_BER_Adaptive

"Calculate_BER_Adaptive" es una función que usa "selected_users" generado por "Scheduler" y la SNR instantánea de cada trama para calcular la BER por usuario. Solo se tiene en cuenta la BER cuando el usuario fue efectivamente seleccionado para transmitir. Se estima la BER con la misma fórmula adaptada a la M usada y se devuelve la BER media por usuario. Para la posterior representación de la PDF de la BER media por usuario.

```
function BER_avg = Calculate_BER_Adaptive(SNR_dB, selected_users)
% Calculate_BER_Adaptive - Calcula la BER esperada usando selección
adaptativa.
% Entradas:
%     SNR_dB          - Matriz SNR instantánea [K x num_tramas]
%     selected_users - Vector con usuario programado por trama
% Salida:
%     BER_avg - BER media estimada por usuario

[K, N] = size(SNR_dB);
ber_sum = zeros(1, K);    % Suma de BERs por usuario
count = zeros(1, K);      % Número de transmisiones por usuario

for n = 1:N
    k = selected_users(n);                      % Usuario seleccionado
    snr_nat = 10^(SNR_dB(k, n) / 10);           % SNR en unidades naturales

    % Selección de modulación y BER asociada
    M = 64;
    ber = 0.2 * exp(-snr_nat / (M - 1));
    if ber >= 1e-3
        M = 16;
        ber = 0.2 * exp(-snr_nat / (M - 1));
    end

    ber_sum(k) = ber_sum(k) + ber;                % Acumular BER
    count(k) = count(k) + 1;
end

BER_avg = ber_sum ./ max(count, 1);            %Realizar la media
end
```

10. Select_Modulation

Para elegir el orden de la modulacion implementamos "Select_Modulation" que evalúa la SNR instantánea en cada trama y calcula la BER estimada usando la fórmula $0.2 \cdot \exp(-\text{SNR} / (M-1))$. Si la BER es menor a 10^{-3} , se

escoge modulación 64QAM, si no, 16QAM. Este proceso genera un vector M_vector con la modulación elegida por trama, optimizando la eficiencia espectral en función de la calidad del canal.

```
function M = Select_Modulation(SNR_dB)
% Select_Modulation – Selecciona entre 64QAM y 16QAM según SNR y BER
% objetivo.
% Entrada:
%     SNR_dB – Valor de SNR instantánea en dBs para un usuario en una
% trama
% Salida:
%     M – Orden de modulación seleccionado (64 o 16)

snr_lin = 10^(SNR_dB / 10); % Convertimos la SNR de dB a lineal

% Primero probamos con 64QAM
M = 64;
ber = 0.2 * exp(-snr_lin / (M - 1)); % Fórmula aproximada de BER

% Si no se alcanza la BER deseada (<10^-3), bajamos a 16QAM
if ber >= 1e-3
    M = 16;
end

end
```

SIMULACIONES

En el apartado a, utilizamos la distribución de los usuarios situados a la misma distancia de 150 metros de la estación base utilizando la función Calc_Pos_MS con la opción circular para generar y posteriormente graficar. El objetivo es analizar el rendimiento en cuanto a retardo y BER bajo diferentes frecuencias Doppler. Se pide el uso de tramas de 32 símbolos y modulación 16QAM fija.

El cálculo inicial de la SNR media para cada usuario se lleva a cabo con la función calcular_SNRs. Estos valores iniciales de SNR representan la media a partir de la cual se simulan los efectos de desvanecimiento rápido (fading) mediante el canal Rayleigh, utilizando la función Apply_Rayleigh_Channel para obtener SNRs instantáneas por símbolo de cada usuario, que varían en el tiempo en función de la frecuencia Doppler.

Se simulan tres casos distintos con frecuencias Doppler de 5 Hz, 50 Hz y 120 Hz, que representan escenarios de baja, media y alta movilidad, respectivamente, y luego los 3 casos anteriores para cada tipo de algoritmo del "Scheduler".

Finalmente, se representa la distribución (PDF) del número de tramas de espera de los usuarios y el total de retardos acumulados por usuario mediante histogramas y gráficos de barras.

En el apartado b, repetimos la simulación del apartado a pero realizando Monte Carlo con 20 distribuciones aleatorias de usuarios con cada frecuencia Doppler y para ambos algoritmos del Scheduler adaptativo. Hemos

representado las CDF del retardo medio, la BER media y la tasa transmitida a cada usuario en las dos distribuciones que tenemos de usuarios para este proyecto.

Y finalmente en el apartado c, realizamos las mismas simulaciones que en el b pero implementamos 3 funciones nuevas como son "Calculate_BER_Adaptive", "Transmit_AdaptiveQAM" y "Select_Modulation". "Calculate_BER_Adaptive" la utilizamos para representar la gráficas de la BER media de los usuarios, teniendo en cuenta que si la BER con 64QAM es menor que 10^{-3} transmitimos con dicha modulación. El resto de funciones mencionadas son empleadas para la transmision donde "Select_Modulation" devuelve las "Ms" que se utilizaran en cada trama y la función "Transmit_AdaptiveQAM" se encargará de la transmision en sí utilizando como parámetros el vector de "Ms" que devuelve "Select_Modulation" y "selected_users" que devuelve "Scheduler".

RESULTADOS

Apartado A)

En el apartado a se generó una distribución circular de 10 usuarios situados a 150 metros de la estación base. La SNR media para cada usuario se calculó mediante la función "calcular_SNRs". Los resultados se representan en un gráfico de barras, observándose cierta variabilidad entre usuarios, predecible debido a la aleatoriedad del canal por shadowing.

También se grafica la PDF del número de tramas de espera (retardo) por usuario. Se simulan tres frecuencias Doppler diferentes (5 Hz, 50 Hz, 120 Hz) tanto con el algoritmo por "utilidad" como con el algoritmo "maxSNR". Con la frecuencia Doppler baja (5 Hz), los retardos tienden a concentrarse en valores más bajos. A una frecuencia media (50 Hz), los retardos aumentan ligeramente por la mayor variación del canal. Y con una frecuencia alta (120 Hz), el canal cambia más rápidamente, y esto produce una mayor aleatoriedad en el scheduling, provocando que esté mas repartido el número de tramas de espera entre usuarios.

Además, se presentaron gráficas de barras con el total de tramas de espera por usuario, mostrando la desigualdad en el acceso al canal bajo cada configuración.

Se muestran el retardo medio y BER media. Utilizando modulación 16QAM sin codificación, y simulando con 960,000 símbolos (30,000 tramas), se calcularon los valores promedio de la BER para cada frecuencia Doppler. Como resultado, comprobamos que a frecuencias Doppler más bajas, la BER tiende a ser menor, ya que el canal es más estable y permite decisiones más acertadas en el scheduling. A medida que aumenta la frecuencia Doppler, la variabilidad del canal impacta negativamente en la precisión del scheduler, lo que eleva la BER. El retardo medio aumenta también con la frecuencia Doppler, especialmente en el caso del algoritmo maxSNR, donde usuarios con malas condiciones de canal tienen un mayor retardo para transmitir.

Comparando ambos algoritmos, el de "maxsnr" maximiza la eficiencia espectral pero provoca que si hay mucha diferencia entre las SNRs de los usuarios, para frecuencias Doppler bajas hace que los usuarios con peor SNR no transmitan. En el caso de frecuencias Doppler más altas, estas diferencias a la hora de qué usuario transmite disminuyen, ya que al tener una mayor variabilidad temporal, los usuarios tienen mayores posibilidades de poder llegar a transmitir. El algoritmo de "utilidad" favorece esa mejor equidad entre usuarios, porque prioriza no solo la calidad del canal, sino que también el tiempo que ha pasado desde la última transmisión. Esto iguala el número de transmisiones entre los usuarios con mejor y peor SNR.

Apartado B)

Siguiendo con el apartado b realizamos un Monte Carlo con 20 distribuciones aleatorias de usuarios en un área de 800x800. Para cada distribución, utilizamos tres frecuencias Doppler (5 Hz, 50 Hz y 120 Hz), aplicando ambos algoritmos del "Scheduler": "utilidad" y "maxsnr". Guardando en variables acumuladoras: los retardos (número de tramas de espera), la BER media (usando 16QAM sin codificación) y la tasa transmitida.

Respecto al CDF del retardo medio por usuario, el algoritmo de "utilidad" genera una mayor igualdad en cuanto al acceso al canal por los usuarios. Esto se refleja en CDF más uniformes. "MaxSNR" en cambio favorece en su mayoría a los usuarios con mejores canales, generando un aumento en los retardos medios de los usuarios con menor SNR. El efecto aumenta con frecuencias Doppler altas, donde el canal varía rápidamente y los algoritmos deben adaptarse con mayor velocidad.

En cuanto a la CDF de la BER media por usuario, "Utilidad" consigue una BER media más baja para la mayoría de los usuarios, gracias a un acceso más regular al canal. "maxSNR", al priorizar usuarios con SNR alta, deja a otros con peores condiciones fuera del scheduling por más tiempo, lo que incrementa su BER media. A frecuencias Doppler altas (120 Hz), ambos algoritmos se ven mas afectados, pero la desigualdad en "maxSNR" es mucho mayor.

Finalmente con la CDF de la tasa transmitida por usuario, "MaxSNR" proporciona tasas más elevadas para unos pocos usuarios, provocando que el resto apenas transmitan viéndose claramente en la CDF un salto brusco. "Utilidad", presenta menor tasa máxima, pero distribuye mejor los recursos, logrando una curva de CDF más progresiva y equilibrada.

APARTADO C)

Este último apartado incorpora la opción para los usuarios de elegir entre dos modulaciones, 16-QAM y 64-QAM, con la condición de tener una BER inferior a 10^{-3} . Al igual que en los apartados anteriores se analiza el comportamiento de un sistema TDMA utilizando distintas frecuencias doppler y algoritmos para la elección de usuarios.

Como se ha observado anteriormente, cuando se utiliza una frecuencia doppler baja, el canal es más estático, por tanto si usamos el algoritmo de máxima SNR los usuarios con mejor canal tienen prioridad, esto provoca una menor BER pero mayor retardo para algunos usuarios. Cuando utilizamos el de 'utilidad' los usuarios se alternan de manera más repartida, pero la eficiencia es menor. Cuando aumentamos aumentamos la frecuencia doppler la desigualdad con el algoritmo de máxima SNR sigue existiendo, aunque de una manera menos exagerada y 'utilidad' sigue siendo la mejor opción si buscamos homogeneidad. En las gráficas podemos observar lo anterior, cuando utilizamos el criterio de máxima SNR hay mas saltos y desigualdades entre los usuarios en comparación con el otro algoritmo.

La distintas opciones de modulaciones que se le ofrecen al usuario resultan en una mayor tasa de bits ya que se está utilizando el espectro de una manera más óptima, siempre manteniendo la BER en el nivel necesario.

En comparación con el apartado anterior la mayor diferencia a destacar es la incorporación de la modulación adaptativa como hemos mencionado antes. Esto se realiza mediante la función Select_Modulation, que prueba con 64-QAM, y en el caso de no cumplir con los requisitos de BER máxima se utiliza 16-QAM. Esto resulta en una mayor tasa de transmisión de bits.

Métrica	Modulación fija (apartado b)	Modulación adaptativa (apartado c)
---------	------------------------------	------------------------------------

Tasa media	Moderada	Aumenta considerablemente
BER media	Aceptable (16QAM)	Similar o mejor (controlada)
Retardo medio	Equitativo con utilidad	Se mantiene con utilidad
Disparidad (maxSNR)	Alta en tasa y retardo	Aún mayor, pero más tasa