Prestazioni Modulazioni

Francesco Musto 0612707371
Andrea Savastano 0612707904
Mario Zito 0612708073

November 17, 2024

Contents

1 Costellazioni Modulazioni

Delle 4 modulazioni prese in esame nelle simulazioni (PSK, PAM, QAM, PPM) ne vengono stampate le costellazioni, ove possibile, nel caso di M=16 ed E_{av}=1.

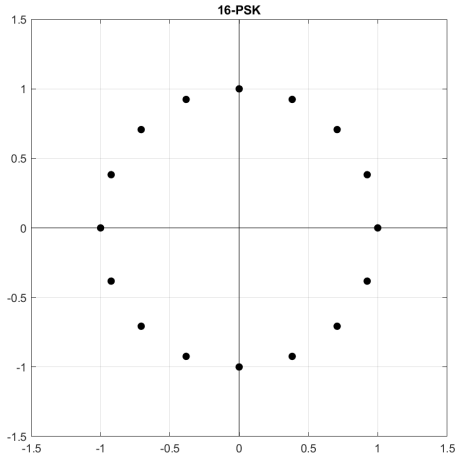


Figure 1: 16-PSK

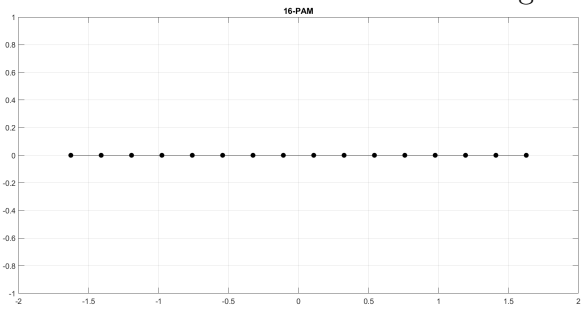


Figure 2: 16-PAM

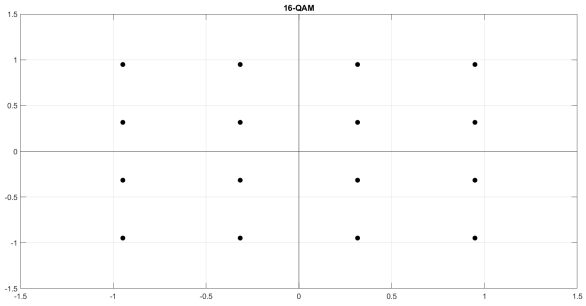
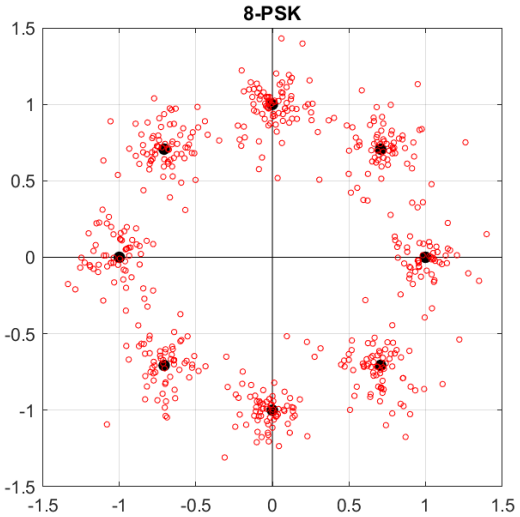
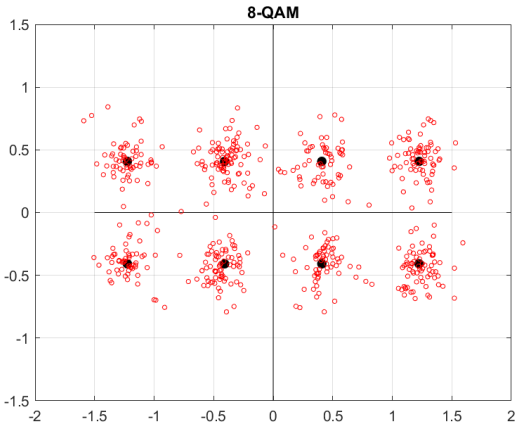


Figure 3: 16-QAM

1.1 Costellazioni con segnali ricevuti

Prendendo in considerazione le modulazioni PSK e QAM con M=8, E_{av}=1, SNR_{dB}=[10:20] e 50 prove MonteCarlo, ne vengono stampate le costellazioni con i segnali ricevuti in rosso.



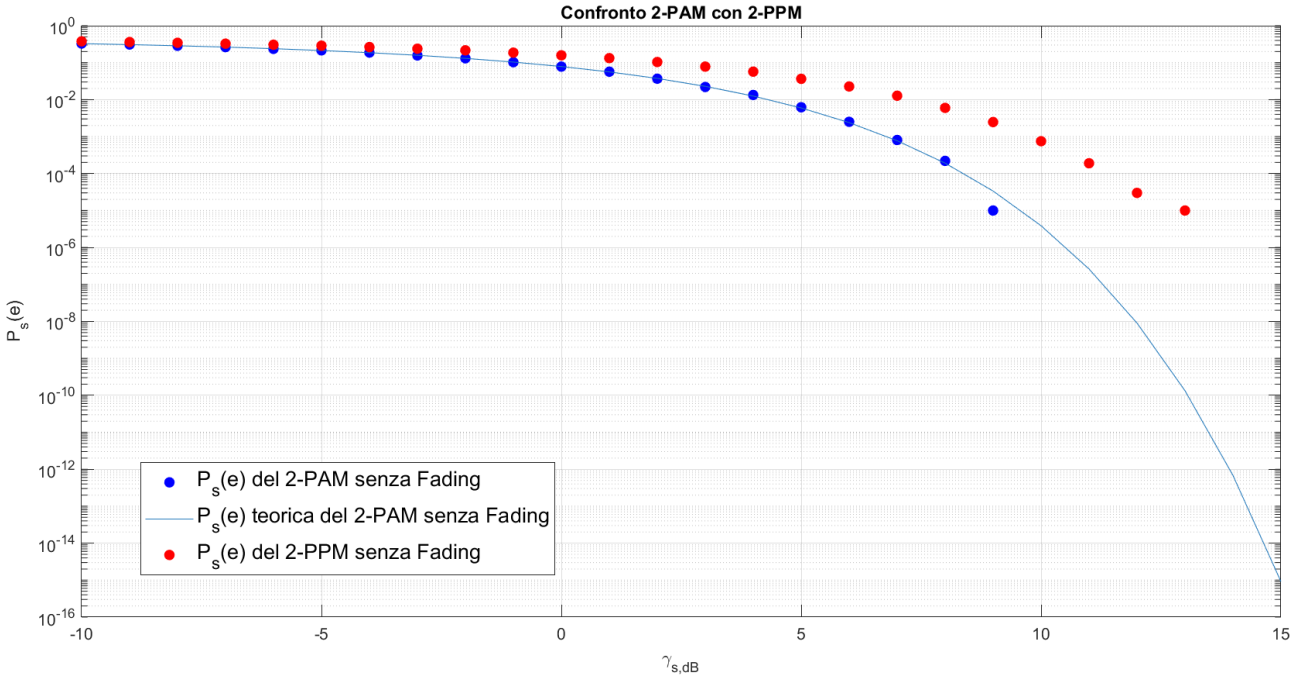


Si noti che i segnali ricevuti si trovano vicino a quelli teorici, grazie all'SNR_{dB} alto.

2 Confronto PAM - PPM

2.1 2-PAM con 2-PPM

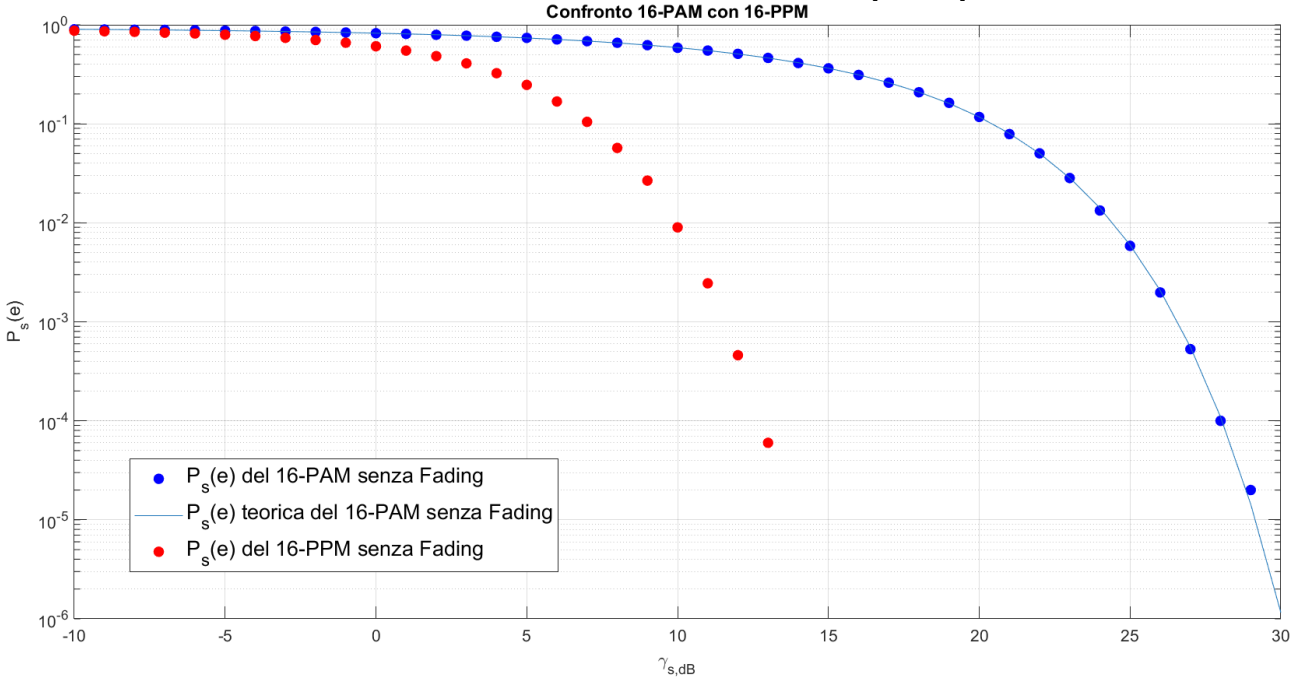
Simulazione delle modulazioni 2-PAM e 2-PPM con SNR_{dB}=[-10:15], 10^5 prove MonteCarlo.



In questa prima simulazione la curva blu, relativa alle prestazioni del 2-PAM, decresce più velocemente rispetto alla curva rossa del 2-PPM, al crescere dell'SNR_{dB}. Questo dimostra che per il numero di bit scelto (k=2), il PAM risulta più prestante del PPM.

2.2 16-PAM con 16-PPM

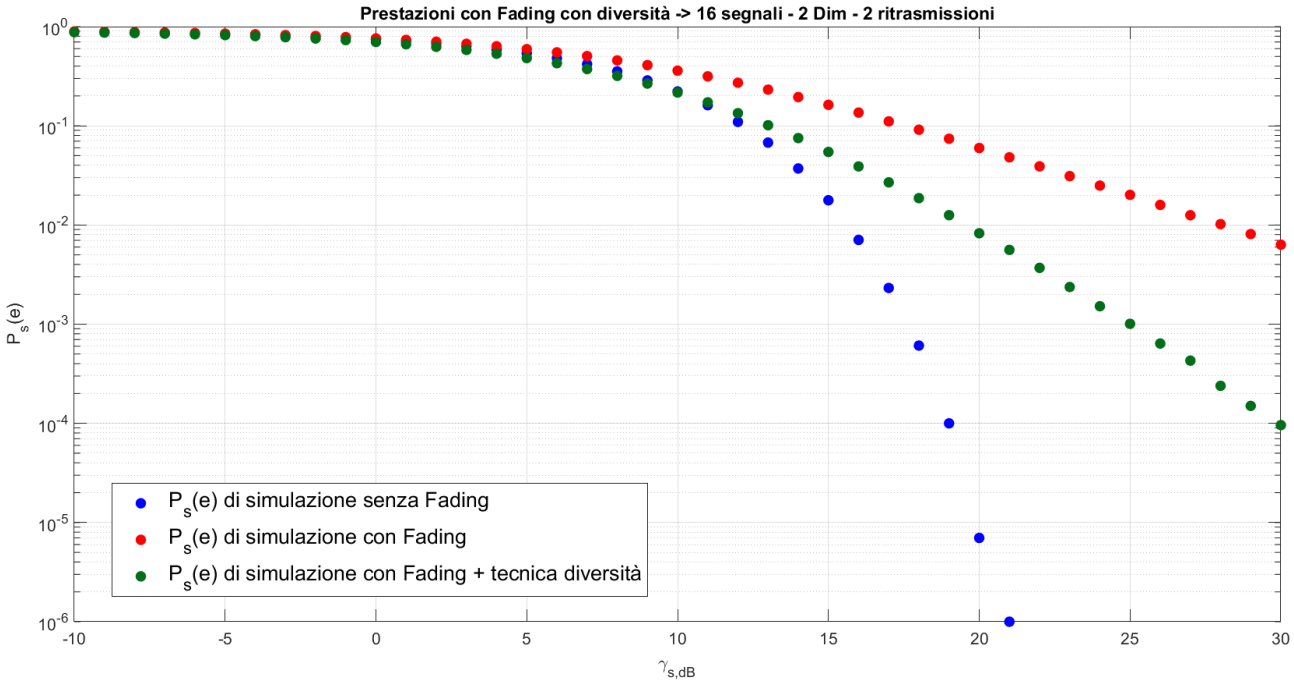
Simulazione delle modulazioni 16-PAM e 16-PPM con SNR_{dB}=[-10:30], 10^5 prove MonteCarlo.



Rispetto alla simulazione precedente, il PPM risulta molto più prestante del PAM per il numero di bit scelto (k=4). Se si continuasse ad aumentare il numero k di bit in ulteriori simulazioni, si noterebbe che le prestazioni del PAM continuerebbero a peggiorare, mentre quelle del PPM a migliorare.

3 Prestazioni 16-QAM

Simulazione di una modulazione 16-QAM con SNR_{dB}=[-10:30], 10^6 prove MonteCarlo e L=2 ritrasmissioni



3.1 Senza Fading

La curva azzurra mostra come la $P_s(e)$ decresce esponenzialmente all'aumentare dell'SNR considerato.

3.2 Con Fading

La curva rossa mostra come la $P_s(e)$ decresce iperbolicamente (più lentamente) all'aumentare dell'SNR.
Si noti come le prestazioni risultano pessime rispetto al caso in cui il Fading è assente.

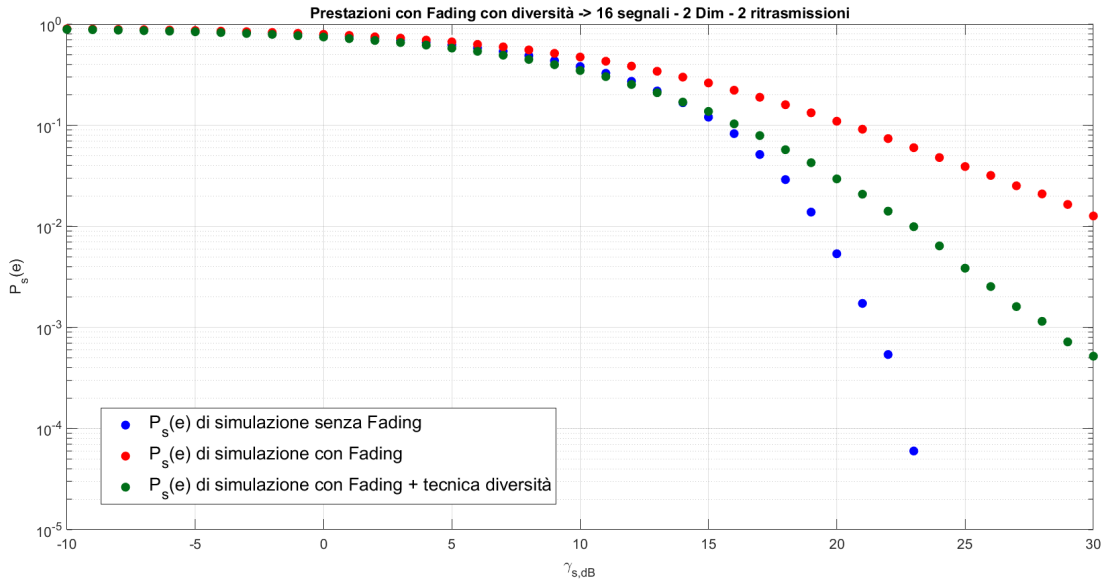
3.3 Con Tecnica di Diversità in tempo/frequenza

La curva verde mostra come la $P_s(e)$ decresce in maniera più veloce rispetto a quella in presenza di Fading.
Utilizzando L=2 ritrasmissioni si riesce a fronteggiare il Fading, ottenendo delle prestazioni tollerabili.

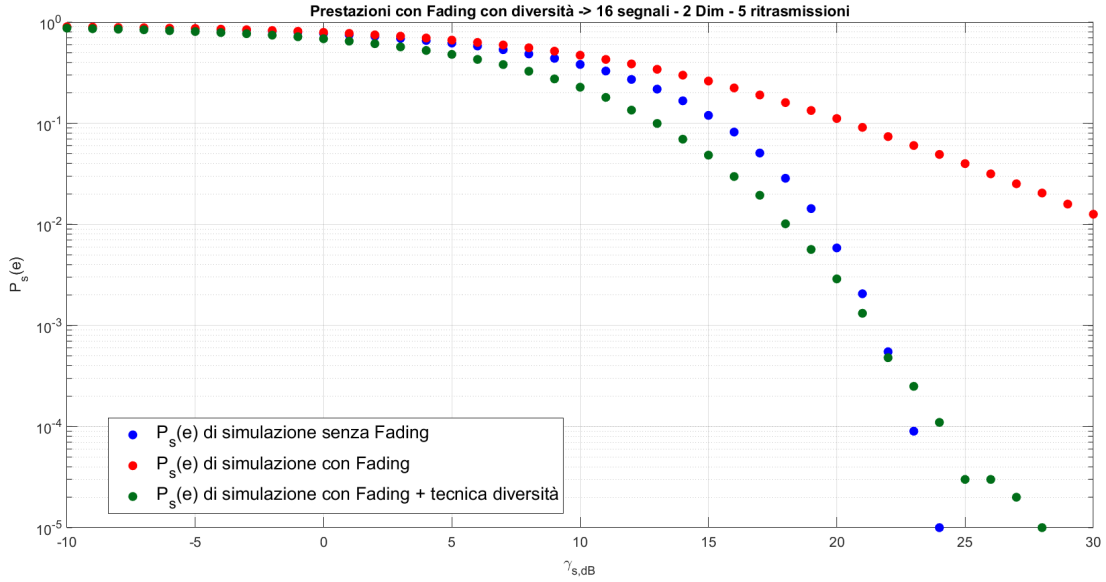
4 Prestazioni 16-PSK al variare di L

Simulazione di una modulazione 16-PSK con SNR_{dB}=[-10:30], 10^5 prove MonteCarlo e ripetuta due volte: la prima con un numero L di ritrasmissioni pari a 2 e la seconda con L=5.

4.1 L=2



4.2 L=5



Confrontando le curve verdi e blu dei 2 grafici è possibile notare che all'aumentare delle ritrasmissioni (da L=2 a L=5) le prestazioni migliorano in maniera significativa.

Entrando nel dettaglio, all'aumentare dell'SNR, la $P_s(e)$ nel primo grafico (L=2) segue inizialmente quella del caso senza Fading per poi allontanarsi da quest'ultima, rimanendo comunque al di sotto di quella con il Fading.

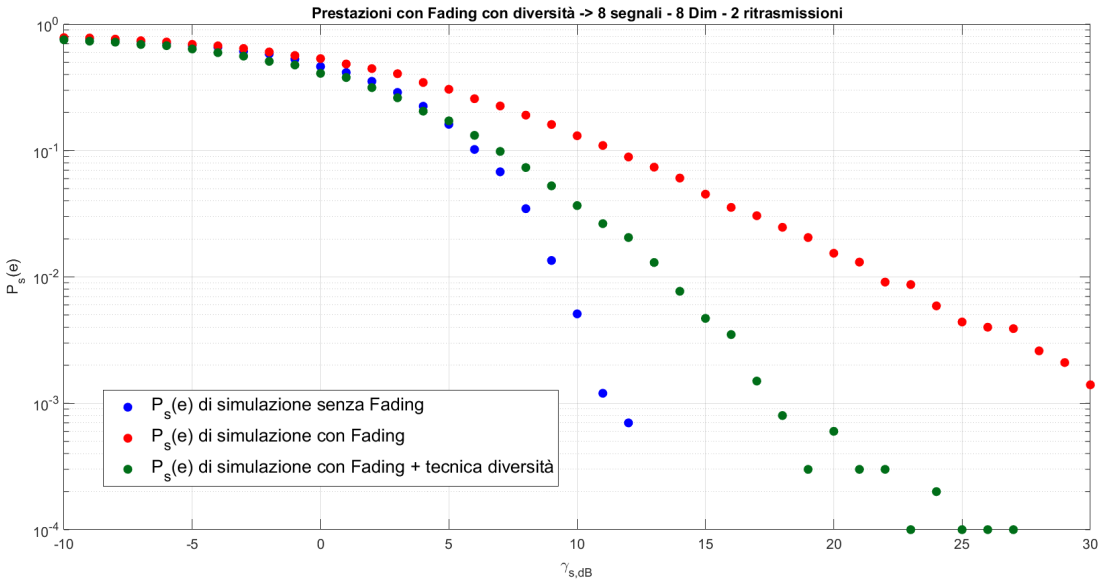
Questo è dovuto al fatto che la tecnica di diversità scelta faccia selezionare tra le L=2 ritrasmissioni quella con maggiore SNR.

Nel secondo grafico (L=5) si può notare che la $P_s(e)$ si trova addirittura al di sotto di quella del caso in cui il fenomeno del Fading non è considerato; sinonimo del fatto che le prestazioni sono migliorate ancor di più.

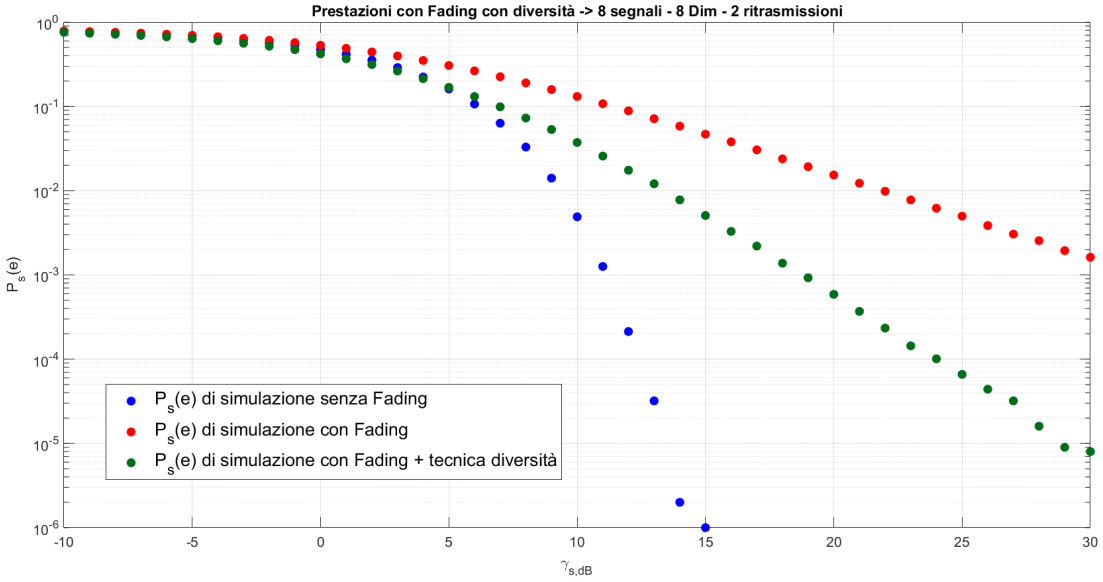
5 Prestazioni 8-PPM al variare di MC

Simulazione di una modulazione 8-PPM con SNR_{dB}=[-10:30], L=2 e ripetuta due volte: la prima con un numero MonteCarlo di prove pari a 10^4 e la seconda con MC=10^6.

5.1 MC=10^4



5.2 MC=10^6



Osservando la prima figura (MC=10^4), la curva azzurra non presenta stime precise della $P_s(e)$ per valori di SNR_{dB} più alti, in particolare da SNR_{dB}=10 fino a 30.
Questo è dovuto al numero insufficiente di prove MC, sulle quali è stata stimata una $P_s(e)$=0 per gli SNR_{dB} alti. In questo caso la $P_s(e)$ non si può mostrare perchè è in scala logaritmica.

Invece per la curva rossa vengono mostrate le stime della $P_s(e)$ per ogni SNR_{dB} perchè, in presenza del Fading (SNR come variabile aleatoria esponenziale), l'SNR effettivo è minore e questo facilita la stima della $P_s(e)$, perchè vi sono più errori.

Osservando la seconda figura (MC=10^6), la curva azzurra presenta stime più precise della $P_s(e)$, fornendone una fino a SNR_{dB}=15. Dopo questo valore, anche in questo caso, non è possibile ottenere stime della $P_s(e)$ perchè il numero MC di prove risulta ancora insufficiente.

6 Resoconto finale

Per effettuare un'analisi complessiva delle simulazioni, si evince che un aumento del rapporto segnale-rumore (SNR) nelle trasmissioni conduce a un miglioramento delle prestazioni.
Tuttavia, tale incremento risulta insufficiente per ottenere prestazioni accettabili in presenza di Fading, rendendo pertanto necessaria l'introduzione di almeno una tecnica di diversità. Quella implementata nelle simulazioni prevede la ritrasmissione dello stesso segnale, consentendo di migliorare le prestazioni anche in condizioni di Fading.

Appendice: Codice Matlab

A Main

```
function proj_main(Str, k, SNRdB, MC, flag, L)
%% parametri In
% --INPUT--
% Str:   e' la modulazione da utilizzare
% k:     numero di bit da trasmettere
% SNRdB: Rapporto segnale rumore per decibel per simbolo
% MC:    numero MonteCarlo di trasmissioni per ogni SNR
% flag:  a 1 per simulare il Fading
% L:     numero di ritrasmissioni per la tecnica di diversita' ...
    (flag=1)

%% Controllo
if k==0
fprintf('k deve essere maggiore di 0\n')
return;
end

%% Scelta modulazione
if strcmpi(Str,'PAM')
Cost = proj_PAM_generator(k);
elseif strcmpi(Str,'PPM')
Cost = proj_PPM_generator(k);
elseif strcmpi(Str,'PSK')
Cost = proj_PSK_generator(k);
elseif strcmpi(Str,'QAM')
Cost = proj_QAM_generator(k);
else
fprintf('Str deve essere PAM / PPM / PSK / QAM \n')
return;
end

%% Simulazione con o senza Fading
proj_estimate_Pe(SNRdB, Cost, MC);
if flag
proj_estimate_Pe_Fading(SNRdB, Cost, MC);
if L>1
proj_estimate_Pe_diversity(SNRdB, Cost, MC, L);
end
end
```

B Generatore costellazioni

B.1 Generatore PAM

```
function Cost = proj_PAM_generator(k)
%% parametri In-Out
% --INPUT--
% k:     numero di bit da trasmettere

% --OUTPUT--
% Cost:  Costellazione dei segnali in forma matriciale -> Mx1 (PAM)

%% calcolo matrice Cost
M = 2^k;
Am = linspace(-(M-1), M-1, M);
Eg = 3/(M^2-1);
Cost = (Am*sqrt(Eg))';

%% stampa costellazione PAM
plot(Cost,zeros(1,M),'ko-','MarkerSize',6,'MarkerFaceColor','k')
hold on
title(M+'-PAM')
grid on
```

B.2 Generatore QAM

```
function Cost = proj_QAM_generator(k)
%% parametri In-Out
% --INPUT--
% k:     numero di bit da trasmettere

% --OUTPUT--
% Cost:  Costellazione dei segnali in forma matriciale -> Mx2 (QAM)

%% calcolo matrice Cost
M = 2^k;
qam_symbols = qammod(0:M-1, M);
Cost(:,1) = real(qam_symbols);
Cost(:,2) = imag(qam_symbols);

Eav = sum(Cost(:,1).^2+Cost(:,2).^2)/M;
Cost = Cost./sqrt(Eav);

%% stampa costellazione PSK
plot(Cost(:,1),Cost(:,2),'ko','MarkerSize',6,'MarkerFaceColor','k')
hold on
plot([-1.5 1.5],[0 0],'k-','MarkerSize',6,'MarkerFaceColor','k')
plot([0 0],[-1.5 1.5],'k-','MarkerSize',6,'MarkerFaceColor','k')
title(M+'-QAM')
grid on
```

B.3 Generatore PSK

```
function Cost = proj_PSK_generator(k)
%% parametri In-Out
% --INPUT--
% k:     numero di bit da trasmettere

% --OUTPUT--
% Cost:  Costellazione dei segnali in forma matriciale -> Mx2 (PSK)

%% calcolo matrice Cost
M = 2^k;
D = M/2;
Cost = zeros(M,2);

% Avendo posto Eav=1 i segnali sono equienergetici: Es=1 per ogni ...
   m --> r=1
% quindi gli M segnali sono equidistanti dal centro della ...
   circonferenza (r=1)
% sx ed sy sono le coordinate dei singoli segnali su x e su y e ...
   quindi
% gli elementi di ogni riga della costellazione Cost.
for ii=1:M
sx = cos(ii*pi/D);
sy = sin(ii*pi/D);
Cost(ii,1) = sx;
Cost(ii,2) = sy;
end

%% stampa costellazione PSK
plot(Cost(:,1),Cost(:,2),'ko','MarkerSize',6,'MarkerFaceColor','k')
hold on
plot([-1.5 1.5],[0 0],'k-','MarkerSize',6,'MarkerFaceColor','k')
plot([0 0],[-1.5 1.5],'k-','MarkerSize',6,'MarkerFaceColor','k')
title(M+'-PSK')
grid on
axis('square')
```

B.4 Generatore PPM

```
function Cost = proj_PPM_generator(k)
%% parametri In-Out
% --INPUT--
% k:     numero di bit da trasmettere

% --OUTPUT--
% Cost:  Costellazione dei segnali in forma matriciale -> MxM (PPM)

%% calcolo matrice Cost
%tutti gli M segnali sono equienergetici e Es = 1 per ogni m
%quindi Eav = = Esm = 1
M = 2^k;
Cost = eye(M);
```

C Stima $P_s(e)$

C.1 $P_s(e)$ senza Fading

```
function Pe_s = proj_estimate_Pe(SNRdB, Cost, MC)
%% parametri In-Out
% --INPUT--
% SNRdB: Rapporto segnale rumore per decibel per simbolo
% Cost:  M righe = M regnali, N colonne = dimensionalita'
% MC:    numero MonteCarlo di trasmissioni per ogni SNR

% --OUTPUT--
% Pe_s:  Probabilita' di errore per simbolo su MC prove e per ...
   diversi SNR

%% parametri utili
SNRnf = 10.^(SNRdB/10); % SNR no Fading per SIMBOLO
Eav = 1;
N0 = Eav./SNRnf; % varia N0
M = length(Cost(:,1));
N = length(Cost(1,:));
Pe_s = zeros(1,length(SNRnf));
%% calcolo Ps(e)
for ii=1:length(SNRnf)
N0_now = N0(ii);
errori = zeros(1,MC);
for jj=1:MC
indexTx = randi(M);
% scegliamo una delle M righe = una degli M segnali
s = Cost(indexTx,:);
r = s + randn(1,N)*sqrt(N0_now/2);

% plot del segnale ricevuto
% if N==1
%     plot(r,0,'ro','MarkerSize',3);
% elseif N==2
%     plot(r(1),r(2),'ro','MarkerSize',3);
% end

% calcolo della minima distanza
d_min = norm(r - Cost(1,:));
indexRx = 1;
for zz=2:M
d_tmp = norm(r - Cost(zz,:));
if(d_tmp < d_min)
d_min = d_tmp;
indexRx = zz;
end
end

errori(jj) = indexTx≠indexRx;
end
Pe_s(ii) = mean(errori);
end

%% Stampa
figure
semilogy(SNRdB, Pe_s, 'bo', 'MarkerSize', 6, 'MarkerFaceColor', 'b')

hold on
```

```
56          title('Prestazioni Modulazione - '+M+' segnali - '+N+' Dim')
57          xlabel('\gamma_{s,dB}')
58          ylabel('P_s(e)')
59          lgd = legend('P_s(e) di simulazione senza Fading');
60
61          if N==1 % la modulazione e' il PAM
62          % non mettiamo log2(M) perche' SNR e' gia' per simbolo
63          Pe_s_th = 2*(M-1)/M * qfunc(sqrt(6/(M^2-1)*SNRnf));
64          semilogy(SNRdB, Pe_s_th, 'Color','#1f77ba');
65          lgd = legend('P_s(e) di simulazione senza Fading', ...
66          'P_s(e) teorica senza Fading');
67          end
68          lgd.FontSize = 13;
69          grid on
```

C.2 $P_s(e)$ con Fading

```
1           function Pe_s = proj_estimate_Pe_Fading(SNRdB, Cost, MC)
2           %% parametri In-Out
3           % --INPUT--
4           % SNRdB: Rapporto segnale rumore per decibel per simbolo
5           % Cost:  M righe = M regnali, N colonne = dimensionalita'
6           % MC:    numero MonteCarlo di trasmissioni per ogni SNR
7
8           % --OUTPUT--
9           % Pe_s:  Probabilita' di errore per simbolo su MC prove e per ...
                diversi SNR
10
11          %% parametri utili
12          SNRnom = 10.^(SNRdB/10);
13          Eav = 1;
14          M = length(Cost(:,1));
15          N = length(Cost(1,:));
16          Pe_s = zeros(1,length(SNRnom));
17          %% calcolo Ps(e)
18          for ii=1:length(SNRnom)
19          errori = zeros(1,MC);
20          for jj=1:MC
21          SNRrv = myexprnd(SNRnom(ii),1,1);
22          N0_now = Eav/SNRrv;
23
24          indexTx = randi(M);
25          s = Cost(indexTx,:);
26          r = s + randn(1,N)*sqrt(N0_now/2);
27
28          d_min = norm(r - Cost(1,:));
29          indexRx = 1;
30          for zz=2:M
31          d_tmp = norm(r - Cost(zz,:));
32          if(d_tmp < d_min)
33          d_min = d_tmp;
34          indexRx = zz;
35          end
36          end
37
38          errori(jj) = indexTx≠indexRx;
39          end
40          Pe_s(ii) = mean(errori);
41          end
```

```
42
43          %% Stampa
44          semilogy(SNRdB, Pe_s, 'ro', 'MarkerSize', 6, 'MarkerFaceColor', 'r')
45
46          title('Prestazioni con Fading '+M+' segnali - '+N+' Dim')
47          lgd = legend('P_s(e) di simulazione senza Fading', ...
48          'P_s(e) di simulazione con Fading');
49
50          if N==1
51          % Calcolo Pe teorica con Fading per il PAM
52          Pe_s_th_F = zeros(1,length(SNRnom));
53          for ii=1:length(SNRnom)
54          Pe_s_th_F(ii) = (M-1)/M * (1-sqrt(1/(1+(M^2-1)/(3*SNRnom(ii))) ));
55          end
56          semilogy(SNRdB, Pe_s_th_F, 'Color', '#FF6506')
57          lgd = legend('P_s(e) di simulazione senza Fading', ...
58          'P_s(e) teorica senza Fading', ...
59          'P_s(e) di simulazione con Fading', ...
60          'P_s(e) teorica con Fading');
61          end
62          lgd.FontSize = 13;
```

C.3 $P_s(e)$ con Fading e tecnica diversità

```
1           function Pe_s = proj_estimate_Pe_diversity(SNRdB, Cost, MC, L)
2           %% parametri In-Out
3           % --INPUT--
4           % SNRdB: Rapporto segnale rumore per decibel per simbolo
5           % Cost:  M righe = M regnali, N colonne = dimensionalita'
6           % MC:    numero MonteCarlo di trasmissioni per ogni SNR
7           % L:     numero di ritrasmissioni per la tecnica di diversita'
8
9           % --OUTPUT--
10          % Pe_s:  Probabilita' di errore per simbolo su MC prove e per ...
                diversi SNR
11
12          %% parametri utili
13          SNRnom = 10.^(SNRdB/10);
14          Eav = 1;
15          M = length(Cost(:,1));
16          N = length(Cost(1,:));
17          Pe_s = zeros(1,length(SNRnom));
18          %% calcolo Ps(e)
19          for ii=1:length(SNRnom)
20          errori = zeros(1,MC);
21          for jj=1:MC
22          indexTx = randi(M);
23          s = Cost(indexTx,:);
24
25          SNRrv = myexprnd(SNRnom(ii),1,L);
26          N0_now = Eav/max(SNRrv);
27          r = s + randn(1,N)*sqrt(N0_now/2);
28
29          d_min = norm(r - Cost(1,:));
30          indexRx = 1;
31          for zz=2:M
32          d_tmp = norm(r - Cost(zz,:));
33          if(d_tmp < d_min)
34          d_min = d_tmp;
```

```
35          indexRx = zz;
36          end
37          end
38
39          errori(jj) = indexTx≠indexRx;
40          end
41          Pe_s(ii) = mean(errori);
42          end
43
44          %% Stampa
45          semilogy(SNRdB, Pe_s, 'o','Color', '#006e18','MarkerSize', 6, ...
                'MarkerFaceColor', '#006e18')
46
47          title('Prestazioni con Fading con diversita' -> '+M+' segnali - ...
                '+N+' Dim - '+L+' ritrasmissioni')
48          lgd = legend('P_s(e) di simulazione senza Fading', ...
49          'P_s(e) di simulazione con Fading', ...
50          'P_s(e) di simulazione con Fading + tecnica diversita'');
51
52          if N==1
53          lgd = legend('P_s(e) di simulazione senza Fading', ...
54          'P_s(e) teorica senza Fading', ...
55          'P_s(e) di simulazione con Fading', ...
56          'P_s(e) teorica con Fading', ...
57          'P_s(e) di simulazione con Fading + tecnica diversita'');
58          end
59          lgd.FontSize = 13;
```

C.3.1 myexprnd ausiliaria

```
1           function y=myexprnd(mu,r,c)
2           y=-mu*log(rand(r,c));
```