ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

ANALIZA HEMINGOVOG KODA PRI PRENOSU QAM MODULACIJOM

– Diplomski rad–

Kandidat:

Nevena Stojanović 2010/406

Mentor:

doc. dr Zoran Čiča

Beograd, Jul 2016.

SADRŽAJ

1. UVOD

Modulacija, analogna ili digitalna je proces obrade informacionog signala u signal koji je pogodan za prenos preko odgovarajućeg komunikacionog medijuma. Generički signal nosilac poruke se naziva modulišući signal, a dok se pomoćni periodični signal naziva nosilac. Signal koji predstavlja rezultat obrade se naziva modulisani signal. Proces obrade modulisanog signala u prijemniku sa ciljem da se izdvoji modulišući signal je demodulacija – inverzan procesu obrade signala u predajniku. Na izlazu iz demodulatora se dobija demodulisani signal. Modulator i demodulator se zajedničkim imenom nazivaju modem. Suština projektovanja digitalnih modema je obezbjeđivanje efikasnog prenosa signala i njegove regeneracije iz signala oštećenog usled šuma i drugih nesavršenosti kanala.

Kvadraturna amplitudska modulacija je složeniji modulacioni postupak, koji se koristi kod prenosa TV signala i kod prenosa digitalnih signala. Cilj svakog komunikacionog sistema je prenijeti što veću količinu informacija kroz što uži frekvencijski opseg. Ovom modulacijom se postiže istovremeni prenos dva korisna, modulišuća signala u istom opsegu učestanosti. Mana amplitudske modulacije je slabija otpornost na smetnje. Zbog toga se koristi zaštitno kodovanje. Zaštitni koder povećava redundantnost digitalnog signala dodavanjem neinformacionih bita informacionom sadržaju. Na taj način se povećava otpornost digitalnog signala na uticaj smetnji, odnosno omogućava se detekcija i korekcija pogrešno prenijetih simbola.

U ovom radu će biti realizovana kvadraturna amplitudska modulacija u okviru Matlab softverskog paketa. Realizacija obuhvata i ispitivanje uticaja zaštitnog kodera, primjenom Hemingovog koda (8, 4).

Ostatak rada organizovan je na sledeći način. U drugom poglavlju biće više riječi o kvadraturnoj amplitudskoj modulaciji, kao i o M-arnoj kvadraturnoj modulaciji. U trećem poglavlju je opisana konstrukcija Hemingovog koda (8, 4) pomoću šablona i date su osnovne karakteristike Hemingovih zaštitnih kodova. Četvrto poglavlje se bavi opisom implementacije koda, poređenjem modulacija za različite vrijednosti *M*. U petom poglavlju će biti dati rezultati simulacije, grafici zavisnosti vjerovatnoće greške po bitu od odnosa signal/šum za sve tri modulacije, sa i bez primjene zaštitnog kodera. Na kraju je dat zaključak u kome su izložena završna razmatranja teze.

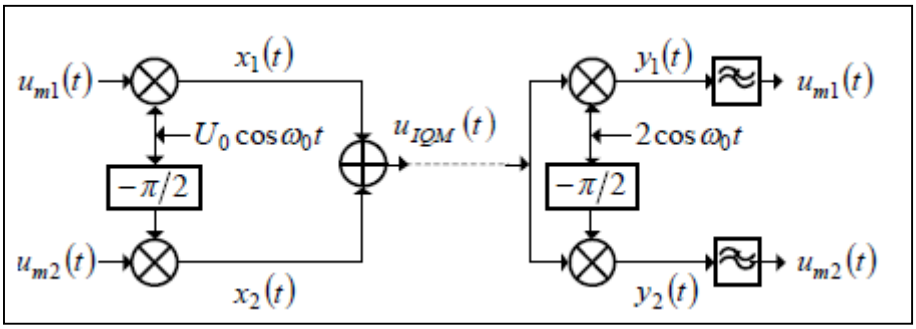
2. KVADRATURNA AMPLITUDSKA MODULACIJA

Modulacioni postupci se mogu podijeliti, generalno, u dvije velike grupe: analogne i digitalne modulacije. Postoje i hibridni modulacioni postupci pri čemu su moguće kombinacije isključivo analognih, isključivo digitalnih, odnosno analogdno/digitalnih modulacionih postupaka.

Kod analognih modulacionih postupaka modulisani signal je kontinualan, odnosno parametri signala nosioca se mijenjaju kontinualno u vremenu u skladu sa vremenskim oblikom datog modulišućeg signala. I u slučaju digitalnih modulacionih postupaka kao nosilac uvijek se koristi signal čiji je talasni oblik sinusoidalan. U odnosu na prethodno opisane postupke modulacije modulišući signal je digitalan, odnosno diskretizovan je u vremenu i po amplitudi.

2.1. Kvadraturna amplitudska modulacija

Kvadraturna amplitudska modulacija ili kako se skraćeno naziva QAM (*Quadrature Amplitude Modulation*) spada u grupu tzv. izvedenih digitalnih modulacija. Generička blok šema u kojoj se koristi kvadraturna modulacija prikazana je na slici 2.1.1.



Slika 2.1.1. Osnovna blok šema sistema sa kvadraturnom modulacijom/demodulacijom [1]

Kvadraturno modulisan signal predstavljen je izrazom

$$u_{IQM}(t) = x_1(t) + x_2(t) = u_{m1}(t)\cos w_0 t + u_{m2}(t)\sin w_o t \qquad (2.1.1.)$$

Pod pretpostavkom da su spektri modulišućih signala $u_{m1}(t)$ i $u_{m2}(t)$ ograničeni istom maksimalnom učestanošću f_m , jasno je da se spektri signala $x_1(t)\ i\ x_2(t)$ poklapaju. Signal na izlazu iz predajnika predstavlja zbir dva signala, $x_1(t)\ i\ x_2(t)$. Ukupan opseg učestanosti koji oni zauzimaju iznosi B=2f_m. Razdvajanje ovih signala iz signala $u_{QAM}(t)$ je moguće jer se koriste nosioci u kvadraturi. Pri tome, niskofrekvencijske komponente signala na ulazima u LPF (*Low Pass Filter*) filtre u prijemniku imaju oblik :

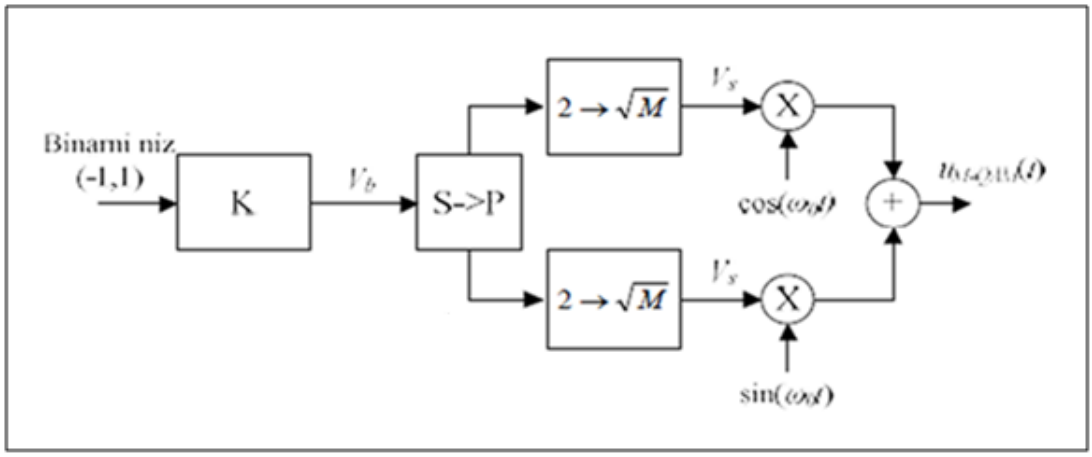
$$\langle y_1(t)\rangle_{LPF} = \langle\, [\, u_{m1}(t)\cos w_0 t + u_{m2}(t)\sin w_o t\,]\, 2\cos w_0 t\, \rangle_{LPF} = u_{m1}(t) \qquad (2.1.2.)$$

$$\langle y_2(t)\rangle_{LPF} = \langle\, [\, u_{m1}\,(t)cos\, w_0 t + \, u_{m2}\,(t)sin\, w_o t\,]\, 2sin\, w_0 t\, \rangle_{LPF} = u_{m2}(t) \qquad (2.1.3.)$$

Koristeći nosioce u kvadraturi postigli smo da se u istom opsegu učestanosti B=2f_m istovremeno prenesu dva korisna, modulišuća signala, što znači da dva puta efikasnije koristimo raspoloživi spektar.

2.2. M-arna kvadraturna amplitudska modulacija

Kod M-arne kvadraturne modulacije (*M-QAM*) umjesto bita prenosimo simbole, tj. umjesto binarnog prenosimo M-arni signal, kao što je prikazano na opštoj blok šemi predajnika datoj na slici 2.2.1.

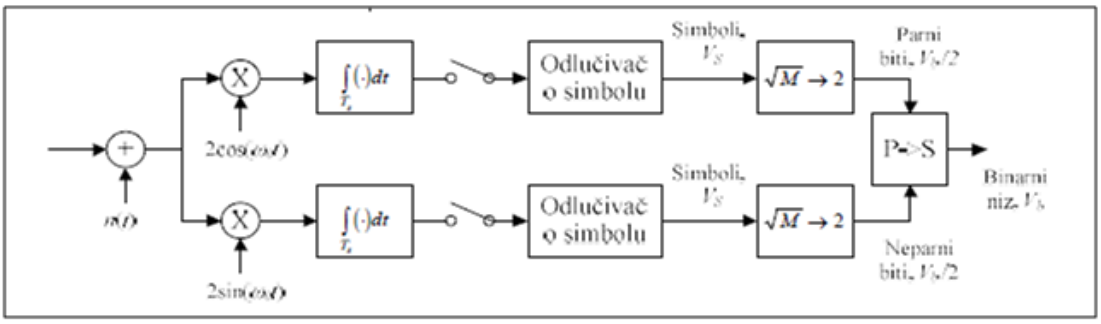


Slika 2.2.1. Opša blok šema predajnika *M-QAM* signala [1]

Blok $2\to\sqrt{M}$ predstavlja konvertor binarnog u M-arni signal. Na ulazu u konvertor u svakoj grani je binarni protok $V_b/2$, a u konvertoru se svaka grupa od *n* bita pretvara u jedan od različitih simbola, pa $\sqrt{M}$ može biti stepen broja 2 tj. *M* može biti stepen broja 4 i imamo da je na linji brzina prenosa

$$V_s = \frac{V_b/2}{\log_2\sqrt{M}} V_s = \frac{V_b/2}{\log_2\sqrt{M}} = \frac{V_b}{\log_2 M} \qquad (2.2.1.)$$

Ovako modulisan signal ide na liniju veze i prolazi kroz kanal koji dodatno unosi šum, pa se posle demodulacije koristi i integrator sa rasterećenjem, što ilustruje slika 2.2.2.



Slika 2.2.2. Opšta blok šema prijemnika *M-QAM* signala [1]

U opštem slučaju M-QAM signal je dat izrazom

$$u\,(t) = \sqrt{\frac{2E_o}{T}}\; a_i \cos w_0 t + \sqrt{\frac{2E_o}{T}}\; b_j \,\mathrm{s}\, in\, w_0 t \qquad (2.2.2.)$$

gdje je $\{a_i, b_j\}$ par nezavisnih cjelobrojnih koeficijenata, čije vrijednosti definišu položaj vrha fazora modulisanog signala u fazorskom prostoru, a E_o je energija *M-QAM* signala najmanje amplitude na jednom signalizacionom intervalu. U praksi se vrijednost *M* bira tako da važi $M = 2^n = 2^{2k},\ n,k\ \in Z$, odnosno stepen modulacije *n* ima vrijednosti n = 2, 4, 6, 8... i tada za fazorski dijagram kažemo da ima kvadratno simetričnu formu, pri čemu u svakom kvadrantu postoji $L = \sqrt{M} = 2^{n/2} = 2^k,\ \ n,k \in Z$, fazora modulisanog signala.

Ako pretpostavimo da su simboli *M-QAM* modulacije jednako vjerovatni, kao i da su grane prijemnika potpuno simetrične dobija se da je srednja energija po simbolu *M-QAM* signala jednaka

$$E_{av} = 2\left[\frac{2E_0}{L}\sum_{i=1}^{L/2}(2i-1)^2\right] = \frac{2(L^2-1)}{3}\,E_0 \qquad (2.2.3.)$$

gdje je $\sum_{i=1}^{L/2}(2i-1)^2 = \frac{L(L^2-1)}{3}$, pa se izraz za *M-QAM* signala može napisati i u obliku

$$u\,(t) = \sqrt{\frac{3E_{av}}{(L^2-1)\,T}({a_i}^2 + {b_i}^2)}\cos\left[w_o t - arctg\,\left(\frac{b_i}{a_i}\right)\right] \qquad (2.2.4.)$$

gdje je $\{a_i, b_j\}$=$\pm 1,\ \pm 3, \ldots,\ \pm(L-1)$ i T =$T_b \log_2 M$ signalizacioni interval *M-QAM* signala.

Vjerovatnoća greške po simbolu na izlazu iz prijemnika M-QAM signala je :

$$P_{e,M-QAM} = 1 - P_c = 1 - (1 - P_{e,q})^2 \cong 2P_{e,q} = 2\frac{L-1}{L}\; erfc\sqrt{\frac{E_o}{N_o}} \qquad (2.2.5.)$$

gdje P_c predstavlja vjerovatnoću tačne odluke, a $P_{e,q}= (1-\frac{1}{L})erfc\sqrt{\frac{E_o}{N_o}}$ vjerovatnoću greške u svakoj od grana prijemnika.

Jednostrana spektralna gustina srednje snage šuma $n(t)$ na ulazu u prijemnik jednaka je N_o. Kada su simboli *M-QAM* signala jednako vjerovatni i kada su grane prijemnika potpuno simetrične dobija se konačni izraz za vjerovatnoću greške po simbolu *M-QAM* signala koji glasi

$$P_{e,M-QAM} = 2\frac{\sqrt{M}-1}{\sqrt{M}}erfc\sqrt{\frac{3E_{av}}{2(M-1)N_0}} \qquad (2.2.6.)$$

što znači da vjerovatnoća greške zavisi i od M i od odnosa signal/šum.

U sledećem poglavlju radu će biti objašnjeni Hemingovi zaštitni kodovi.

3. HEMINGOVI ZAŠTITNI KODOVI

.

Za smanjenje grešaka u telekomunikacionim sisitemima koristi se zaštitno kodovanje. U literaturi zaštitno kodovanje se često označava kao *ECC* (*Error Control Coding*). Da bi se greške nastale u prenosu poruka otkrile i ispravile, moraju se u prenošene poruke uvesti dodatni (redundantni) biti. Ovi redundantni biti mogu se uvesti na različite načine i u zavisnosti od toga kodovi se mogu i podijeliti prema načinu njihovog unošenja na blok kodove i na konvolucione kodove.

Hemingov kod je jedan od najpoznatijih blok kodova. Hemingov kod se obilježava sa (*n, k*), pri čemu je *k* broj informacionih bita, a *n* ukupan broj bita u kodovanoj riječi tj. $n=m+k$, gdje je *m* broj dodatnih zaštitnih bita.

Osnovni Hemingovi kodovi imaju dužinu $n = 2^m - 1$ i takav je kod (7,4). Prošireni Hemingovi kodovi su nastali od originalnih kodova dodavanjem bita parnosti (n+1,k), a skraćeni Hemingovi kodovi, u oznaci (n-j, k-j), se dobijaju ispuštanjem poslednjih informacionih bita prilikom računanja šablona.

3.1. Hemingov kod (8, 4)

Hemingov kod (8, 4) je nastao od originalnog Hemingovog koda (7, 4) dodavanjem bita provjere parnosti. Da bi kodovali četiri informaciona bita u oznaci I_1, I_2, I_3, I_4 potrebno je odrediti pozicije zaštitnih bita Z_1 , Z_2, Z_3 . U tabeli 3.1.1. je prikazan šablon za Hemingov kod (8, 4). Pozicije prve jedinice u kolonama (počev od krajnje desne) određuju pozicije zaštitnih bita. Informacioni biti se stavljaju redom na pozicije između pozicija zaštitnih bita. Preostale jedinice u pojedinim kolonama određuju kontrolne sume:

$$Z_1 = I_1 + I_2 + I_4 \qquad (3.1.1.)$$

$$Z_2 = I_1 + I_3 + I_4 \qquad (3.1.2.)$$

$$Z_3 = I_2 + I_3 + I_4 \qquad (3.1.3.)$$

Provjera na parnost zaštitnih i informacionih bita

$$Z_4 = Z_1 + Z_2 + I_1 + Z_3 + I_2 + I_3 + I_4 \qquad (3.1.4.)$$

predstavlja četvrti zaštitni bit, bit parnosti, koji stavljamo na osmu poziciju u kodnoj riječi. U svim jednačinama od 3.1.1. do 3.1.4 se sabiranje vrši po modulu 2.

Kodna riječ ima sledeću strukturu $c = (Z_1, Z_2, I_1, Z_3, I_2, I_3, I_4, Z_4)$.

Decimalni zapis	Binarni zapis	Struktura kodne riječi
1	001	Z_1
2	010	Z_2
3	011	I_1
4	100	Z_3
5	101	I_2
6	110	I_3
7	111	I_4

Tabela 3.1.1. Šablon za Hemingov kod

Na prijemu se vrši dekodovanje pomoću sindroma. Neka su odgovarajući primljeni biti y_i, gdje *i* uzima vrijednosti od 1 do 7. Na osnovu tabele 3.1.1. formiramo sume:

$$s_1 = y_1 + y_3 + y_5 + y_7 \quad (3.1.5.)$$

$$s_2 = y_2 + y_3 + y_6 + y_7 \quad (3.1.6.)$$

$$s_3 = y_4 + y_5 + y_6 + y_7 \quad (3.1.7.)$$

U sume ulaze biti koji sada odgovaraju pozicijama svih jedinica u odgovarajućim kolonama. Po izračunavanju suma formira se sindrom $S = (s_3, s_2, s_1)$.

Provjera parnosti se vrši na sledeći način

$$s_4 = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 \quad (3.1.8.)$$

U zavisnosti od vrijednosti sindroma i bita provjere parnosti razlikujemo četiri slučaja:

- Kada je i vrijednost sindroma, *S* i *s4* (bita provjere parnosti) jednaka nuli, znači da nije bilo greške pri prenosu ili je kompletna riječ pogrešna tj. svih 8 bita (što je mala vjerovatnoća).
- *S* = 0 i *s4* = 1 znači da se greška desila na osmom bitu, koji služi za provjeru parnosti.
- $S \neq 0$, a *s4*=1 znači da postoji jedna greška na poziciji koju pokazuje sindrom (vrijednost sindroma se interpretira u dekadnom sistemu) ili se desilo 3, 5, 7 grešaka (mala vjerovatnoća).
- $S \neq 0$, a *s4*=0 znači da postoje dvije greške ili se desilo 4 ili 6 grešaka, što je mala vjerovatnoća.

Na ulazu u linearni blok kod je jedan od 2^k mogućih blokova od *k* informacionih simbola, a na izlazu je odgovarajuća kodna riječ dužine *n*, koja je po nekom kriterijumu odabrana od 2^n mogućih kandidata. Osobina linearnog blok koda je da skup njegovih kodnih riječi treba da sadrži kombinacije „sve nule“ i „sve jedinice“, jer su to neutralni elementi u odnosu na sabiranje i množenje i treba da važi zatvorenost u odnosu na sabiranje, dok zatvorenost u odnosu na množenje nije neophodna.

Jedan linearan kod može biti opisan bazom- skupom linarno nezavisnih kodnih vektora. Ovi vektori se mogu složiti u obliku generišuće matrice dimenzija *k*x*n* sa linearno nezavisnim vrstama. Generišuća matrica se označava sa *G* .

$$G = \begin{bmatrix} g_{11} & \cdots & g_{1n} \\ \vdots & \dots & \vdots \\ g_{k1} & \cdots & g_{kn} \end{bmatrix} \tag{3.1.9.}$$

Svaka od kodnih riječi se može dobiti sabiranjem jedne, dvije... ili svih *k* vrsta generišuće matrice, pa je ukupan broj kodnih riječi 2^k.

Zamjenom mjesta pojedinih kolona matrice 3.1.9. ne mijenja se snaga koda, već dobijamo generišuću matricu sistematskog koda (G_s). Generišuća matrica sistematskog koda ima oblik

$$G_s=[I_k \quad P] \tag{3.1.10.}$$

gdje je I_k jedinična matrica dimenzija *k*x*k*, a matrica *P* je generalizovana provjera na parnost, dimenzija *k*x(*n*-*k*).

Oblik kodne riječi kod sistematskog koda je $c = (I_1,\ I_2,\ I_3,\ I_k,\ Z_1\ ,Z_2,\ Z_3,\ Z_{n-k})$ tj. grupisani su informacioni biti.

Analogno generišućoj matrici postoji i kontrola matrica (*H*), koja služi u procesu dekodovanja i mora da zadovolji identitet

$$GH^T = 0 \tag{3.1.11.}$$

U slučaju sistematskog koda

$$H = [-P^T \quad I_{n-k}] \tag{3.1.12.}$$

Da bi se izvršilo dekodovanje prispjelog vektora *y*, treba na prijemu odrediti vektor

$$S = y * H^T \tag{3.1.13.}$$

Jedna kombinacija sindroma odgovara jednom tipu grešaka, a samo u posebnim slučajevima decimalna predstava sindroma pokazuje na poziciju greške.

Hemingovo rastojanje je jednako broju pozicija gdje se dvije sekvence bita iste dužine razlikuju.

Minimalno Hemingovo rastojanje (*d*) je minimalan broj jedinica u sekvenci grešaka koji će dovesti do toga da greška ne bude otkrivena.

Da bi kod mogao da koriguje e_c grešaka, treba da bude zadovoljen uslov

$$d \geq 2e_c + 1 \tag{3.1.14.}$$

Da bi mogao da ispravi e_c grešaka i detektuje e_d grešaka treba da bude

$$d \geq e_c + 1 + e_d \tag{3.1.15.}$$

Broj jedinica u kodnoj riječi naziva se Hemingova težina.

Pošto je svaka kodna riječ zbir dvije druge kodne riječi, rastojanja između pojedinih kodnih riječi mogu se odrediti preko Heminogovih težina.

Broj kodnih riječi sa određenim Hemingovim rastojanjima (težinama) predstavlja spektar kodnih rastojanja datog koda.

U slučaju binarnog koda da bi se , pored svih jednostrukih grešaka (koju mogu da budu na *n* pozicija), ispravilo i e_c grešaka mora da bude ispunjena nejednakost

$$2^{n-k} \geq \sum_{i=1}^{e_c} \binom{n}{i} \qquad (3.1.16.)$$

koja se naziva Hemingova granica. Kodovi koji zadovoljavaju Hemingovu granicu sa jednakošću nazivaju se perfektni kodovi i takav je npr. Hemingov kod (7, 4) za $e_c = 1$.

Hemingov kod (8, 4) ima minimalno Hemingovo rastojanje *d*=4, što znači da ima mogućnost ispravljanja jedostrukih greška i detektovanja dvostrukih grešaka. Ne zadovoljava Hemingovu granicu sa jednakošću, a kodni količnik mu je $R = \frac{k}{n} = \frac{4}{8} = \frac{1}{2}$.

Ako je vjerovatnoća da se desi greška na kodnom bitu *p*, vjerovatnoće da se u kodnoj riječi pojavi 3, 4, 5, 6 grešaka kod Hemingovog koda (8, 4) iznose redom

$$P_{e,3} = \binom{8}{3} p^3(1-p)^5 \qquad P_{e,4} = \binom{8}{4} p^4(1-p)^4$$

$$P_{e,5} = \binom{8}{5} p^5(1-p)^3 \qquad P_{e,6} = \binom{8}{6} p^6(1-p)^2$$

Jedino sa sigurnošću znamo da dekoder ne griješi ako su se desile jednostruke ili dvostruke greške, dok kod svih ostalih grešaka može da izazove grešku prilikom dekodovanja.

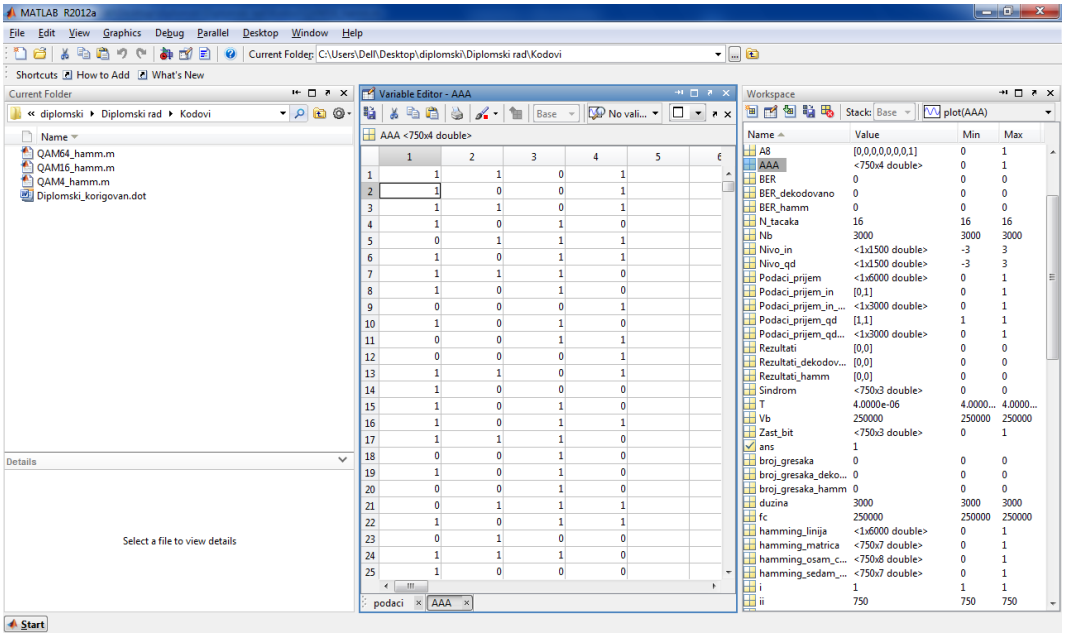
Povećanjem dužine Hemingovog koda raste kodni količnik, ali i vjerovatnoća greške.

4. OPIS IMPLEMENTACIJE KODA

Prvo se generiše slučajan niz nula i jedinica koji predstavlja informacione bite. Informacioni biti se dovode na ulaz Hemingovog kodera (8, 4) gdje se formiraju kodne riječi. Da bi se formirale kodne riječi odrede se zaštitni biti i bit provjere parnosti, koji se skupa sa informacionim bitima smjeste u matricu. Dijeljenjem kodne riječ na dvije sekvence formiramo paralelne nizove, koji će služiti za određivanje nivoa signala grane u fazi i grane u kvadraturi. Množenjem signala sa prostoperiodičnom funkcijom se vrši modulacija. Na prijemu je inverzan postupak tj. radi se demodulacija signala i donošenje odluke. Posle demodulacije određujemo sindrom, odnosno vršimo zaštitno dekodovanje, detekciju i korekciju greške.

Pomoću funkcije `rand([0 1], 1, Nb)` se generiše slučajan niz 0 i 1 od 1 do *Nb*, gdje je *Nb* broj informacionih bita izabran proizvoljno i smjesti se u promjenjivu `podaci`. Koristimo i Monte Karlo simulaciju (for petlja kroz koju propuštamo ceo kod). Monte Karlo metode su stohastičke simulacijske metode, algoritmi koji pomoću slučajnih brojeva i velikog broja proračuna i ponavljanja predviđaju ponašanje složenih matematičkih modela.

Informacija se na osnovu parametara Hemingovog koda , `n_hamming = 8;` `k_hamming = 4;` prikaže u obliku matrice `AAA` čija je struktura prikazana na slici 4.1.



Slika 4.1. Struktura matrice AAA

Svaki red matrice *AAA* predstavlja informacione bite, na osnovu kojih se formiraju zaštitni biti. Kao što se vidi na osnovu tabele 3.1.1. prvi zaštitni bit je suma prvog, drugog i četvrtog

informacionog bita. Drugi zaštitni bit je suma prvog, trećeg i četvrtog informacionog bita, a treći zaštitni bit je suma drugog, trećeg i četvrtog informacionog bita.

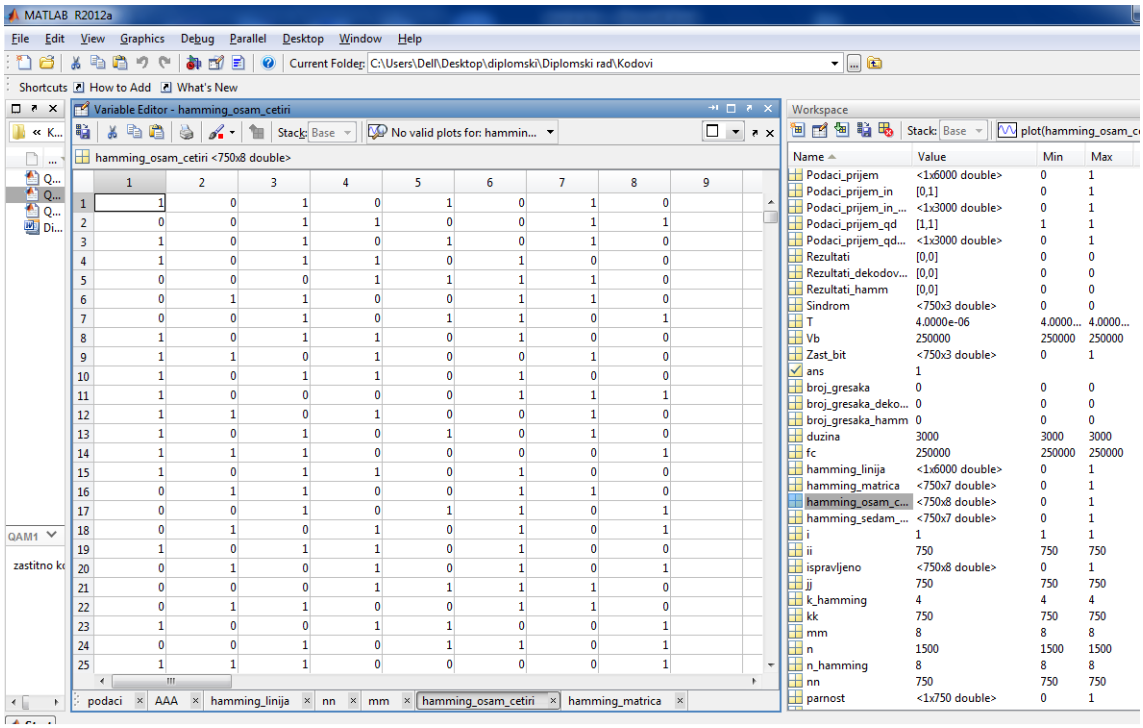
U kodu smo koristili *f*or petlju zbog toga što je broj informacionih bita veći od 4.

```
for ii = 1:Nb/k_hamming
    z1 = mod(AAA(ii,1) + AAA(ii,2) + AAA(ii,4), 2); % odredjivanje prvog zastitnog bita za svaku kodnu rijec
    z2 = mod(AAA(ii,1) + AAA(ii,3) + AAA(ii,4), 2);      % odredjivanje drugog zastitnog bita za svaku kodnu rijec
    z3 = mod(AAA(ii,2) + AAA(ii,3) + AAA(ii,4), 2);      % odredjivanje treceg zastitnog bita za svaku kodnu rijec
    end
hamming_matrica = [AAA Zast_bit];   % spajanje informacionih i zastitnih bita
```

hamming_matrica sadrži informacione i zaštitne bite.

Četvrti zaštitni bit je bit provjere parnosti i dobija se kao suma prvih 7 bita kodne riječi.

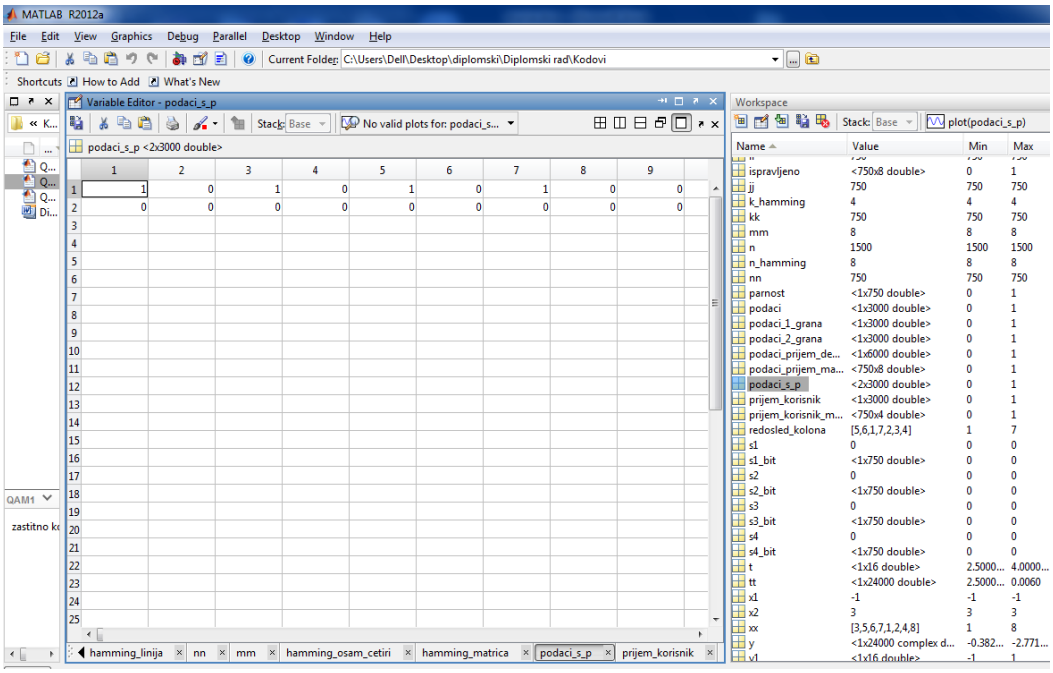
Svaka kodna riječ sadrži po 8 bita i smješena je u matrici *hamming_osam_cetiri*, koja je prikazana na slici 4.2.



Slika 4.2. Struktura matrice *hamming_osam_cetiri*

Na ulaz modulatora stižu biti koji se nalaze u matrici *hamming_linija*, dimenzije 1x2*Nb*.

Kako bi odredili nivoe signala I i Q ose, bite iz matrice *hamming_linija*, podijelimo na dva dijela, koje smjestimo u matricu *podaci_s_p*, koja je prikazana na slici 4.3.



Slika 4.3. Struktura matrice *podaci_s_p*

4-QAM modulacija ima 4 različita stanja i omogućava prenos 2 bita po simbolu. 16-QAM modulacija ima 16 različitih stanja i omogućava prenos 4 informaciona bita po simbolu. 64-QAM modulacija ima 64 različita stanja iomogućava prenos 6 bita po simbolu tj. signalizacionom intervalu.

```
% Primjer odredjivanja nivoa signala I ose kod 16-QAM modulacije
Nivo_in = [];
for n = 1:2:length(podaci_s_p);
   if podaci_s_p(1,n)==0 && podaci_s_p(1,n+1)==0;
      x1=(-3);
   elseif podaci_s_p(1,n)==0 && podaci_s_p(1,n+1)==1;
      x1=(-1);
   elseif podaci_s_p(1,n)==1 && podaci_s_p(1,n+1)==0;
      x1=(1);
   else podaci_s_p(1,n)==1 && podaci_s_p(1,n+1)==1;
      x1=(3);
   end
Nivo_in = [Nivo_in x1];
end
```

Iteracija petlje je sa korak 2 zbog toga što se porede dva susjedna bita prve vrste matrice, *podaci_s_p* dobijene konverzijom serijskog niza u paralelni niz i u zavisnosti da li imaju vrijednosti 00, 01, 10 ili 11 formiraju se nivoi -3, -1, 1 i 3. Analogno je i za Q granu samo se posmatra druga vrsta pomenute matrice.

Modulisani signal (y) je zbir dva signala, signala grane u fazi pomnoženog sa kosinusnom funkcijom i grane u kvadratutri pomnožene sa sinusnom funkcijom. Znači da modulisani signal ima svoj realni i imaginarni deo.

Kanal kroz koji prolazi modulisani signal je modeliran aditivnim belim Gausovim šumom, pomoću funkcije *y_noise = awgn(y,SNR);* gdje je sa SNR (*Signal to Noise*) označen odnos signal/šum.

Na prijemu se radi demodulacija signala, inverzna postupku modulacije.

Takođe se imaginarni i realni deo signala posebno propuštaju kroz integrator sa rasterećenjem. On vrši usrednjavanje (integraciju) signala u vremenskom trajanju simbola čime se značajno smanjuje vjerovatnoća greške. Signali na izlazu iz integratora se porede sa pragovima čije su vrijednosti 2, 0, -2 i donosi se odluka.

U matrici *Podaci_prijem* koja je dimenzija 1x2*Nb* su smješteni biti dobijeni posle bloka za demodulaciju i odlučivač tj. biti na ulazu u Hemingov dekoder. Dekodovanje se radi pomoću sindroma. Prvo se kreira *podaci_prijem_matrica* gdje svaki red predstavlja primljenu kodnu riječ i zatim se određuje sindrom.

```
for jj = 1:Nb/k_hamming
  %%%% odredjivanje sindroma
   s1 = mod(podaci_prijem_matrica(jj,1) + podaci_prijem_matrica(jj,3) + podaci_prijem_matrica(jj,5) + podaci_prijem_matrica(jj,7), 2);
   s2 = mod(podaci_prijem_matrica(jj,2) + podaci_prijem_matrica(jj,3) + podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7), 2);
   s3 = mod(podaci_prijem_matrica(jj,4) + podaci_prijem_matrica(jj,5) + podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7), 2);
   s4 = mod(podaci_prijem_matrica(jj,1) + podaci_prijem_matrica(jj,2) + podaci_prijem_matrica(jj,3) + podaci_prijem_matrica(jj,4) + ...
       podaci_prijem_matrica(jj,5) + podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7) + podaci_prijem_matrica(jj,8),2);
   s1_bit = [s1_bit s1];
   s2_bit = [s2_bit s2];
   s3_bit = [s3_bit s3];
   s4_bit = [s4_bit s4];
end
```

Neka su primljeni biti smješteni u *podaci_prijem_matrica*, prilikom dekodovanja se na osnovu šablona formiraju sume. U sume ulaze biti koji sada odgovaraju pozicijama svih jedinica u odgovarajućim kolonama. Po računanju suma se formira sindrom.

```
Sindrom = [s3_bit' s2_bit' s1_bit'];      % svaki red predstavlja sindrom za odgovarajucu kodnu rijec
```

U kodu se na osnovu pomoćnog vektora A, koji se sabira sa sindromom vrši detekcija i korekcija grešaka.

```
%%%% pomocni vektori A
A1 = [1 0 0 0 0 0 0 0];
A2 = [0 1 0 0 0 0 0 0];
A3 = [0 0 1 0 0 0 0 0];
A4 = [0 0 0 1 0 0 0 0];
A5 = [0 0 0 0 1 0 0 0];
A6 = [0 0 0 0 0 1 0 0];
A7 = [0 0 0 0 0 0 1 0];
A8 = [0 0 0 0 0 0 0 1];


 %%%%% korekcija i detekcija gresaka
ispravljeno = podaci_prijem_matrica;
for kk = 1:Nb/k_hamming
   if Sindrom(kk,:) == [0 0 1];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A1, 2);
   elseif Sindrom(kk,:) == [0 1 0];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A2, 2);
   elseif Sindrom(kk,:) == [0 1 1];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A3, 2);
   elseif Sindrom(kk,:) == [1 0 0];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A4, 2);
   elseif Sindrom(kk,:) == [1 0 1];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A5, 2);
   elseif Sindrom(kk,:) == [1 1 0];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A6, 2);
   elseif Sindrom(kk,:) == [1 1 1];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A7, 2);
   elseif (Sindrom(kk,:) == [0 0 0]) & (s4_bit(kk) == [1]);
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A8, 2);end
```

Funkcija *reshape* mijenja dimenzije matrice i formira matricu *podaci_prijem_dekodovano* koja predstavlja kodne riječi nakon eventualne korekcije tj. prije izvlačenja informacionih bita.

```
podaci_prijem_dekodovano = reshape(ispravljeno',1,(Nb/k_hamming)*8);
```

Postavljanjem kolona na svoje mjesto i izvlačenjem informacionih bita smo omogućili da do korisnika stižu informacioni biti smješteni u matrici *prijem_korisnik*.

Na kraju se na osnovu ugrađene Matlab funkcije računa BER (*Bit Error Rate*) vjerovatnoća greške po bitu i broj grešaka za tri slučaja.

U prvom slučaju se porede biti na ulazu u modulator tj. kodna riječ (*hamming_linija*) sa sekvencom na izlazu iz demodulatora (*Podaci_prijem*) i određuje se broj grešaka i vjerovatnoća greške.

```
[broj_gresaka_hamm(i),BER_hamm(i)]=biterr(hamming_linija,Podaci_prijem);
% mjerenje BER-a i broja bitskih gresaka na prijemu prije zas.dekodovanja
```

U drugom slučaju se porede biti na ulazu u modulator tj. kodna riječ (*hamming_linija*) sa sekvencom *podaci_prijem_dekodovano* koja predstavlja kodne riječi nakon eventualne korekcije tj. prije izvlačenja informacionih bita.

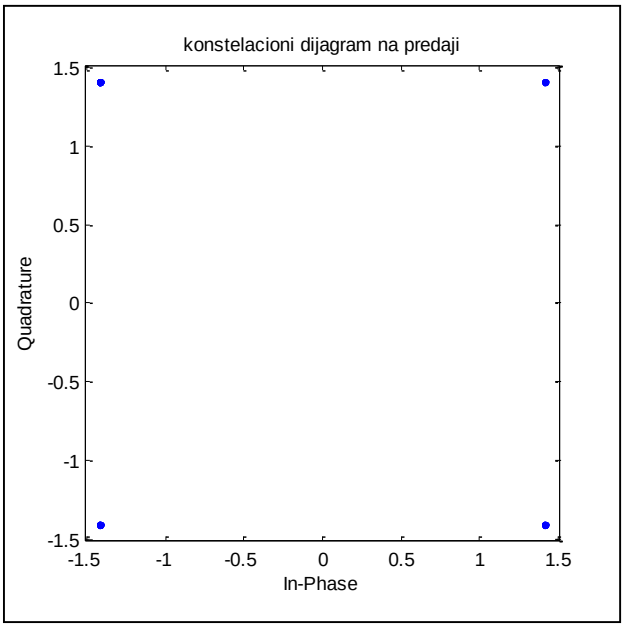
```
[broj_gresaka_dekodovano(i),BER_dekodovano(i)]=biterr(hamming_linija,podaci
_prijem_dekodovano);  % mjerenje BER-a i broja bitskih gresaka na prijemu
nakon dekodovanja
```

U trećem slučaju se porede informacioni biti na ulazu u Hemingov koder (*podaci*) sa informacionim bitima koji dolaze do korisnika (*prijem_korisnik*).

```
[broj_gresaka(i),BER(i)] = biterr(podaci,prijem_korisnik); % mjerenje BER-a i
broja bitskih gresaka na prijemu posle zas. dekodovanja
```

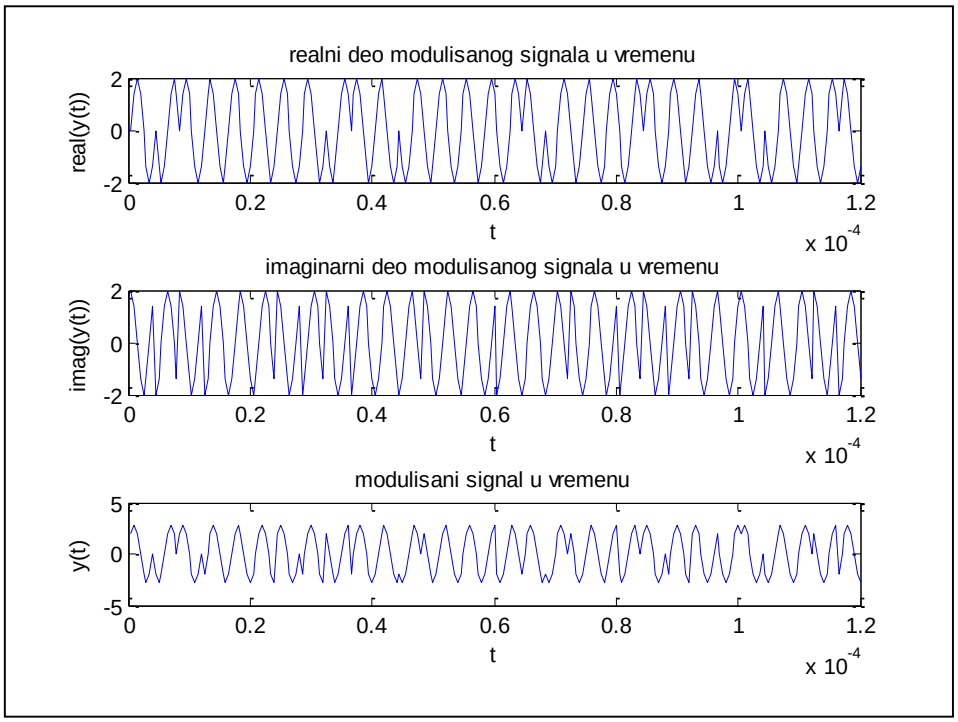
4.1. 4-*QAM* modulacija

Opšti izraz za *M-QAM* signal sastoji se od dvije komponente, kao što se iz jednačine (2.2.2) može vidjeti. Prva komponenta je komponenta u fazi *(*cos*), a druga komponenta je komponenta u kvadraturi (**sin*), pa je konstelacioni dijagram dvodimenzionalan, jer imamo I i Q granu. Ovdje je riječ o konstelacijama tačaka, jer pričamo o digitalnim signalima koji mogu uzimati samo diskretne vrijednosti. Razlog zašto imamo dvodimenzionalnu konstelaciju je jer se *M-QAM* signal predstavlja pomoću kompleksnih brojeva (I i Q grana), a pošto je svaki kompleksan broj (fazor) u konstelaciji određen amplitudom i fazom, onda je konstelacija dvodimenzionalna. Na slici 3.1.1. je prikazan konstelacioni dijagram na predaji za 4-*QAM*.



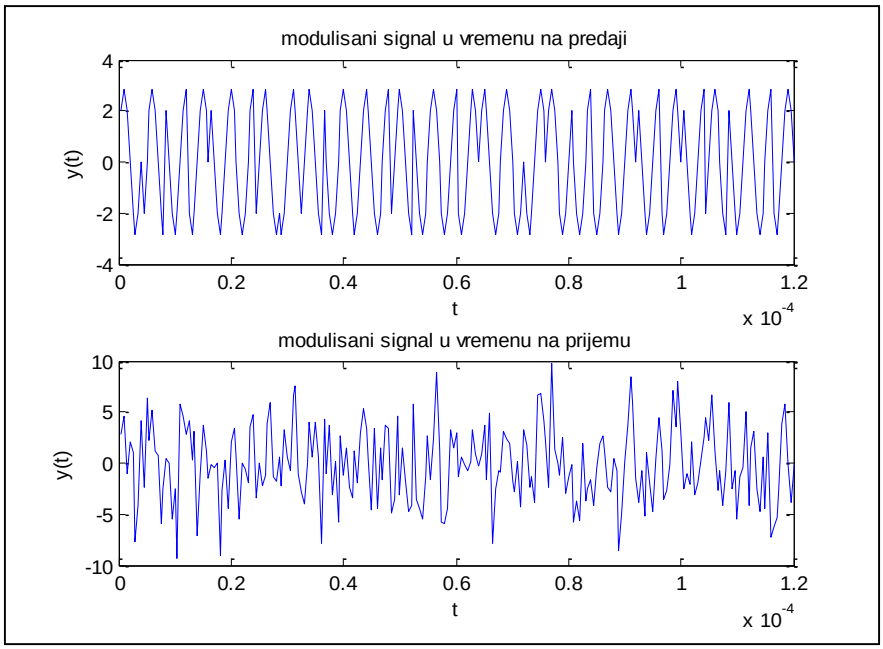
Slika 3.1.1. Konstelacioni dijagram na predaji za 4-*QAM*

Simulacijom u Matlabu se dobija slika 3.1.2. na kojoj je prikazan modulisan signal u vremenu (njegov realni i imaginarni deo).



Slika 3.1.2. Modulisani signal u vremenu za 4-*QAM*

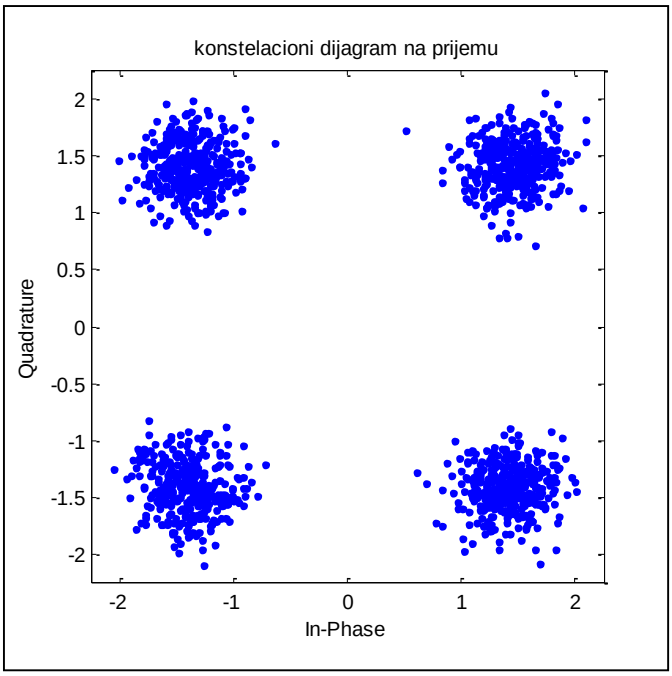
Modulisani signal na prijemu se razlikuje od modulisanog signala na predaji, jer na liniji veze imamo šum, a i kanal sam po sebi unosi šum, što je ilustrovano na slici 3.1.3.



Slika 3.1.3. Modulisani signal na predaji/prijemu za 4-*QAM*

Konstelacioni dijagram na prijemu izgleda kao na slici 3.1.4.

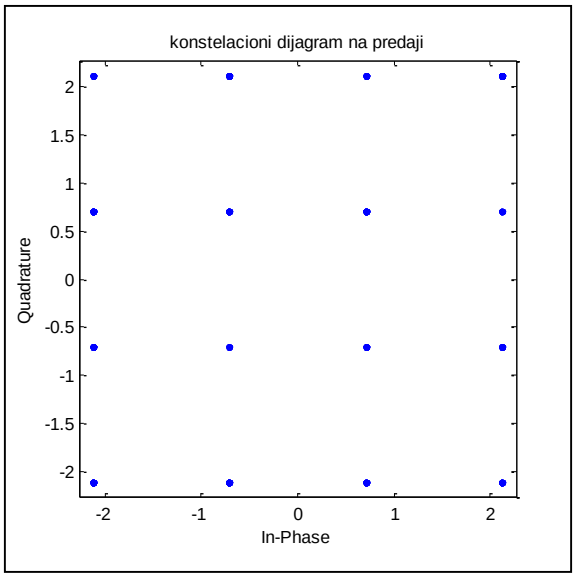
Usled šuma tačke počinju da odstupaju od osnovnih tačaka prikazanih na slici 3.1.1. Što je odnos signal/šum nepovoljniji odstupanje tačaka postaje sve veće i u jednom momentu one postaju bliže susjednim osnovnim tačkama i tada dolazi do greške u procesu demodulacije.



Slika 3.1.4. Konstelacioni dijagram na prijemu za 4-*QAM*

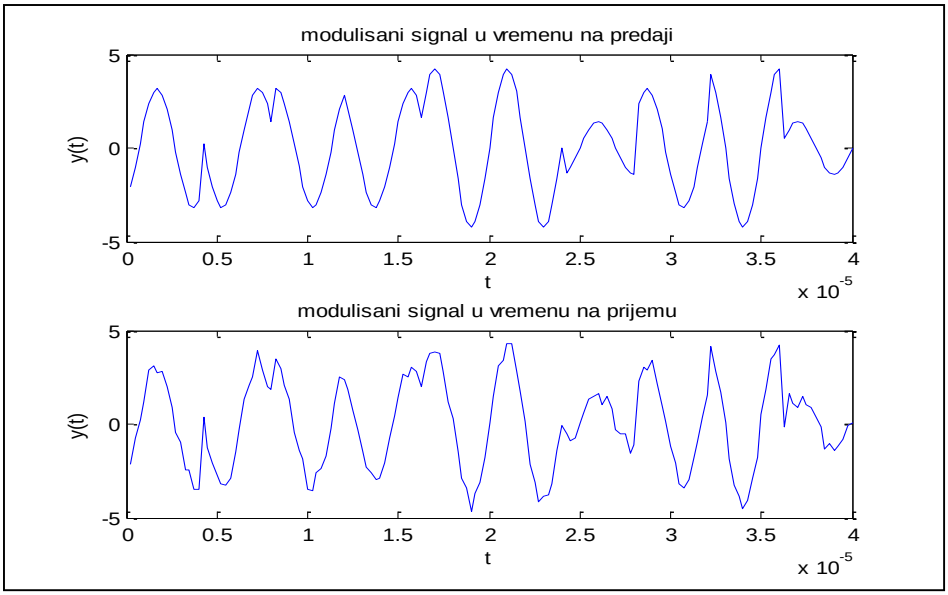
4.2. 16-*QAM* modulacija

Sada je M=16, pa će se konstelacioni dijagram razlikovati u odnosu na 4-QAM modulaciju, kao što je prikazano na slici 3.2.1.



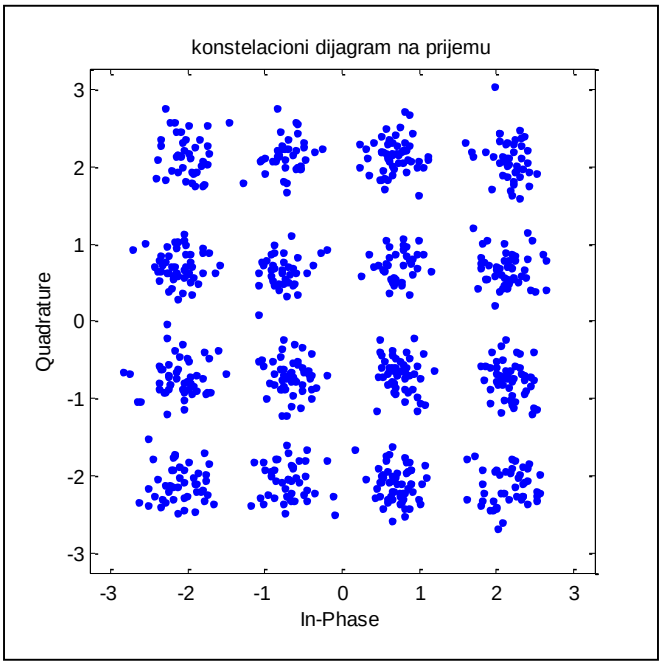
Slika 3.2.1. Konstelacioni dijagram na predaji za 16-*QAM*

Kod modulisanog signala na predaji/prijemu se može uočiti promjena faze, kao i promjena amplitude.



Slika 3.2.2. Modulisani signal na predaji/prijemu za 16-*QAM*

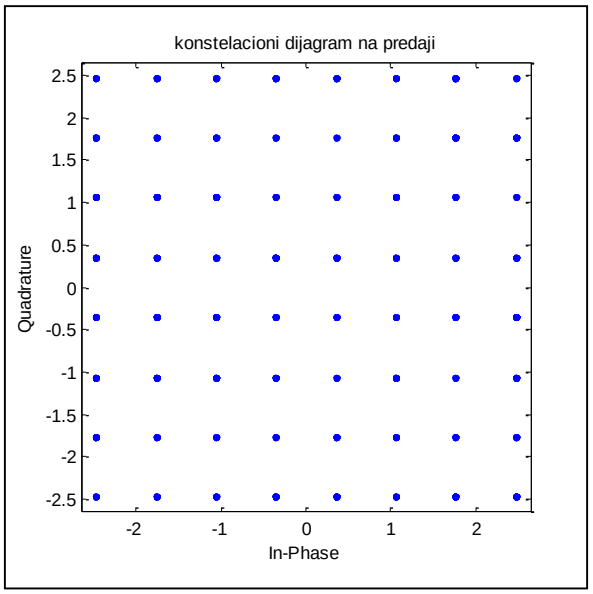
Konstelacioni dijagram na prijemu je dat na slici 3.2.3.



Slika 3.2.3. Konstelacioni dijagram na prijemu za 16-*QAM*

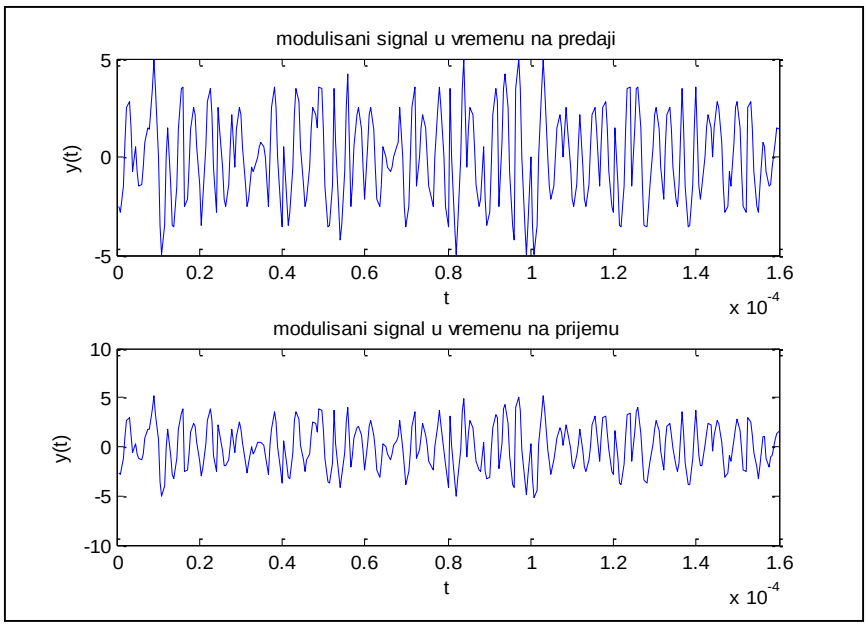
4.3. 64-*QAM* modulacija

64-QAM modulacija ima 64 različita stanja i konstelacioni dijagram je prikazan na slici 3.3.1.



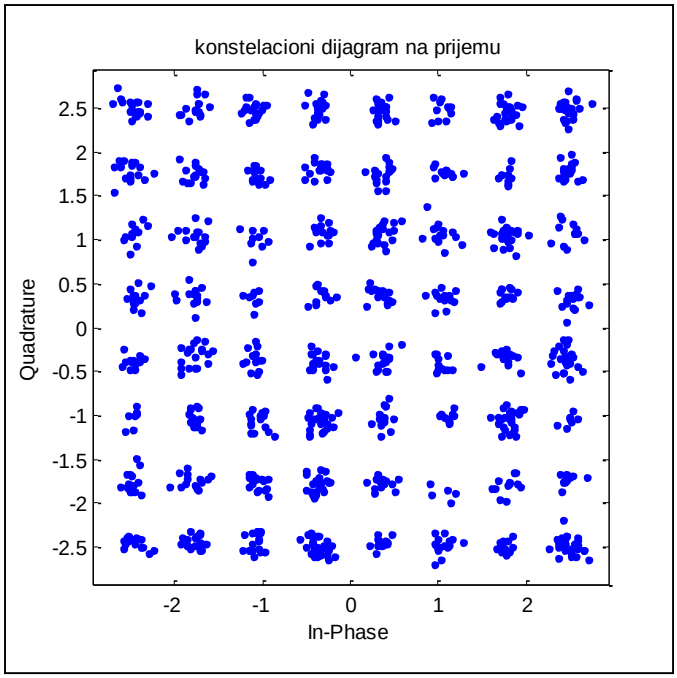
Slika 3.3.1. Konstelacioni dijagram na predaji za 64-*QAM*

Na slici 3.3.2. su prikazani modulisani signali u vremenu na predaji i na prijemu



Slika 3.3.2. Modulisani signal u vremnu na predaji i prijemu za 64-*QAM*

Što je odnos signal/šum manji, šum će da zaguši signal i konstelacioni dijagrami na predaji na prijemu će znatno da se razlikuju, tj. prilično je šetanje tačaka po konstelacionom dijagramu. Konstelacioni dijagram 64-*QAM* signala na prijemu je prikazan na slici 3.3.3.



Slika 3.3.3. Konstelacioni dijagram an prijemu za 64-QAM

U narednom poglavlju ćemo uraditi poređenje sve tri modulacije sa stanovišta vjerovatnoće greške za isti odnos signal/ šum

Poredićemo 4-QAM, 16-QAM i 64-QAM modulacije i to slučajeve kada se ne koristi zaštitni koder i kada imamo zaštitni koder.

5. POREĐENJE *QAM* MODULACIJA

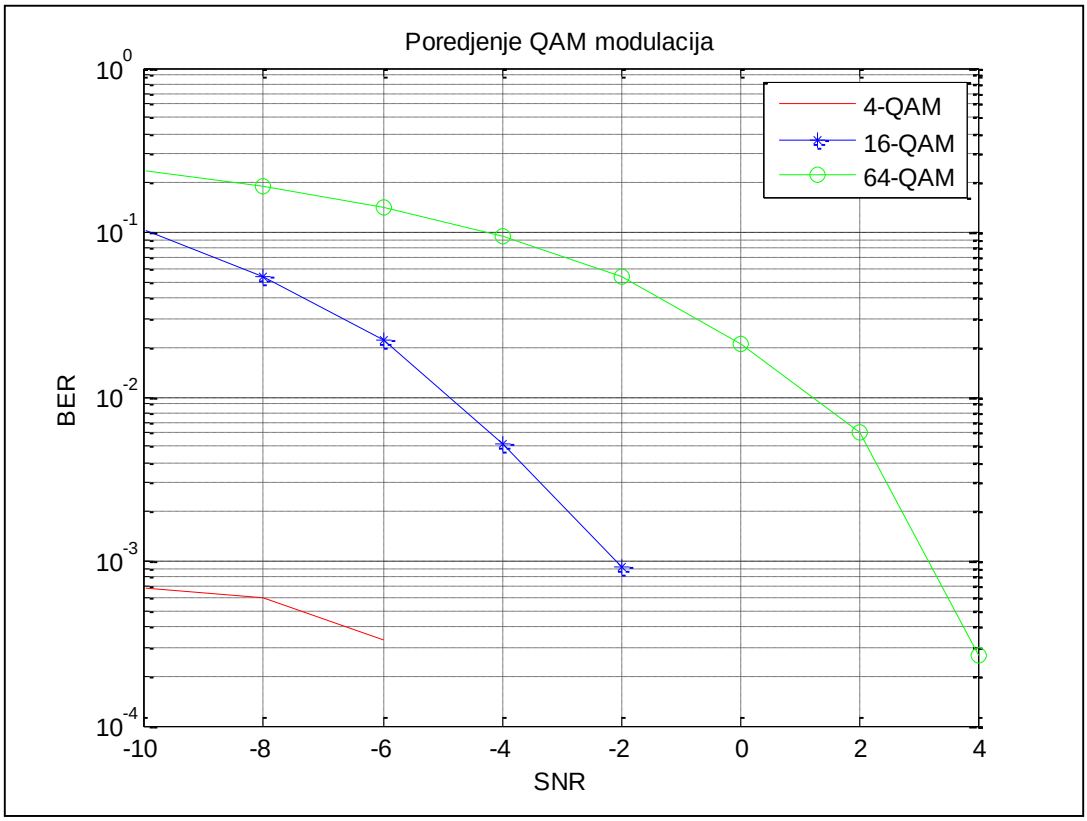
5.1. Poređenje *QAM* modulacija bez primjene zaštitnog kodera

U tabeli su prikazane vrijednosti vjerovatnoće greške po bitu, BER za 4-*QAM*, 16-*QAM* i 64-*QAM* modulacije za isti odnos SNR, dobijene simulacijom.

SNR (dB)	**BER**		
	4-QAM	***16-QAM***	***64-QAM***
-10	0.006867	0.102933	0.2362
-8	0.0006	0.054133	0.189067
-6	0.000333	0.0222	0.141333
-4	0	0.0052	0.093733
-2		0.000933	0.053133
0		0	0.021133
2			0.0006
4			0.000267
6			0

Tabela 4.1.1. Zavisnost BER-a od odnosa SNR

Na slici 4.1.2. je tabela ilustrovana i grafički.



Slika 4.1.2. Vrijednost BER-a za različite vrijednosti odnosa signal/šum za 4-*QAM*, 16-*QAM* i 64-*QAM*

Zaključujemo da sa porastom *M*, raste i vjerovatnoća greške po bitu, tj. lošija je energetska efikasnost. Npr. primjenom 16-*QAM* postupka modulacije zahtijeva se ld*M*=ld16=4 puta uži propusni opseg sistema(npr. u odnosu na *BPSK* modulaciju), odnosno 4 puta se poveća spektralna efikasnost, ali je mana što se tako povećava i vjerovatnoća greške po bitu.

Takođe za manji odnos *SNR* modulacija sa manjom vrijednošću *M* daje i bolji rezultat po pitanju greške, pa kada imamo veliki šum treba koristiti što manju konstelaciju.

5.2. Poređenje *QAM* modulacija kada se koristi Hemingov koder

Osnovni postupak zaštite od grešaka u toku prenosa sastoji se od detekcije greške i ispravke greške. Blok za zaštitni kod se dodaje prije bloka za modulaciju signala, a na prijemnoj strani se prvo uradi demodulacija, pa dekodovanje signala.

U ovoj tezi se koristi Hemingov kod (8, 4), koji je nastao proširenjem osnovnog Hemingovog koda (7, 4) tako što je dodat bit provjere parnosti.

Hemingov kod (8, 4) četiri informaciona bita mapira u osam kodnih.

U tabelama 5.3.1, 5.3.2. i 5.3.3. je prikazana zavisnost vjerovatnoće greške po bitu za različite vrijednosti odnosa signal/šum. U prvoj koloni je prikazana vrijednost BER-a prije zaštitnog kodovanja, tj. vrijednost dobijena poređenjem kodne riječi sa sekvencom na izlazu iz demodulatora. Druga kolona predstavlja vrijednost BER-a dobijenu poređenjem kodne riječi sa sekvencom koja predstavlja kodne riječi nakon eventualne korekcije tj. prije izvlačenja informacionih bita. Vrijednost BER-a poslije zaštitnog dekodovanja je prikazana u trećoj koloni (porede se info biti na ulazu sa info bitima koji stižu do korisnika).

SNR (dB)	**BER**		
	Prije zaštitnog dekodovanja	***Nakon dekodovanja***	***Posle zaštitnog dekodovanja***
-10	0.007	6.6667e-4	6.6667e-4
-8	0	0	0

Tabela 5.2.1. *BER* u funkciji *SNR* za 4-*QAM*

SNR (dB)	**BER**		
	Prije zaštitnog dekodovanja	***Nakon dekodovanja***	***Posle zaštitnog dekodovanja***
-10	0.0965	0.0810	0.0736
-8	0.0527	0.0372	0.0379
-6	0.0192	0.0123	0.0133
-4	0.0063	0.0038	0.004
-2	0.0013	0.0012	6.6667e-4
0	0	0	0

Tabela 5.2.2. *BER* u funkciji *SNR* za 16-*QAM*

SNR (dB)	BER		
	Prije zaštitnog dekodovanja	***Nakon dekodovanja***	***Posle zaštitnog dekodovanja***
-10	0.2278	0.2253	0.2313
-8	0.1912	0.1812	0.1850
-6	0.1487	0.1337	0.1437
-4	0.0985	0.0822	0.0873
-2	0.0492	0.0408	0.0433
0	0.0185	0.0142	0.0160
2	0.0063	0.0035	0.0033
4	5e-4	5e-5	3.333e-4
6	0	0	0

Tabela 5.2.3. Vrijednosti *BER*-a za različite vrijedenosti odnosa signal/šum za 64-*QAM*

Zaključujemo da se primjenom Hemingovog koda (8,4) smanjuje vjerovatnoća greške u odnosu na slučaj kada se ne koristi zaštitni kod. Uvođenjem dodatnih redundantnih bita u prenošene poruke su se greške nastale u prenosu poruke otkrile i ispravile. Hemingov kod (8,4) spada u grupu proširenih linearnih blok kodova. Znači ovim kodom možemo da otkrijemo i ispravimo jednu grešku i da otkrijemo dvije greške. Naziv blok kod je dobio po tome što posmatra prenošenje podataka u blokovima. Zadatak blok kodera je da prihvati određeni broj (*k*) bita i i da ih predstavi određenom kodnom riječi dužine *n* bita. Broj *n* mora biti veći od broja *k*, jer ispravljanje grešaka zahtijeva postojanje dodatnih bita. Blok kodovi se označavaju sa *(n,k)*. A količnik *k* i *n* se naziva kodni količnik koda i u ovom slučaju iznosi ½.

Dekoder je implementiran tako da sve ispravlja po sindromu bez obzira na broj grešaka, pa pokušavajući da ispravi grešku može da poveća broj grešaka.

6. ZAKLJUČAK

M-QAM modulacija ima široku primjenu. 4G mobilne mreže koje predstavljaju budućnost javnih mobilnih mreža koriste 16-QAM modulaciju, baš zbog visoke spektralne efikasnosti, a nekada i 64-QAM modulaciju, jer omogućava veiliki protok krajnjim korisnicima (325 Mbps).

Kada je riječ o mobilnim sistemima i ograničenosti radio spektra od *M-QAM* modulacija se najčešće koriste: *4-QAM, 16-QAM i 64-QAM* modulacioni postupci. Osnovna prednost im je korišćenje *M*-arnog signaliziranja umjesto binarnog. Kombinovanjem nekoliko uzastopnih binarnih simbola (‘0’ i ‘1’) dobijamo *M*-arne simbole. Trajanje *M*-arnog simbola, u slučaju kombinovanja *n* uzastopnih binarnih simbola je $T_M = nT_b$, gdje je T_b trajanje jednog bita. Drugim riječima protok *M*-arnog signala je *n* puta manji u odnosu na binarni signal , što znači da je za prenos takvog signala potreban manji propusni opseg, pa *M*-arno signaliziranje tj. *M*-arni prenos ima veću spektralnu efikasnost. Cijena koja se plaća je da za istu srednju snagu imamo veću vjerovatnoću greške korišćenjem *M*-arnog prenosa u odnosu na binarni prenos. Da bi se pri povećanju *M* zadržala ista vjerovatnoća greške po bitu, potrebno je povećati srednju snagu na predaji ili smanjiti protok.

Djelimično rešenje ovog problema je primjena zaštitnog kodovanja. Međutim prikazani Hemingov kod je namijenjen za usamljene (pojedinačne) greške koje mogu nastati usled pojave Gausovog šuma. Postoje sredine u kojima se javljaju greške koje nisu statistički nezavisne, već se javljaju u paketima (razni vidovi impulsnih smetnji). Tu se može desiti da uzastopni biti budu pogrešni (npr. 4 uzastopna bita su pogrešna). Za ispravljanje svih grešaka morao bi se koristiti Hemingov kod sa velikim brojem zaštitnih bita, što je nepraktično ili čak nemoguće. U ovakvim slučajevima treba dodatno koristiti i interliving.

Interliving se radi na taj način što se , konkretno u ovom slučaju (gdje se očekuju paketi grešaka dužine 4), riječi ne šalju redom, već se uzimaju 4 uzastopne kodne riječi, pa se šalju samo prvi biti od sve četiri riječi , potom samo drugi biti itd. Na prijemu se vrši deinterliving tj. riječi se vraćaju u originalan redosled, pa se onda vrši dekodovanje. Na ovaj način smo postigli da ako se u prenosu desi greška dužine četiri, na prijemu će biti razbijena na četiri pojedinačne greške u četiri različite kodne riječi, a to kod (8, 4) može uspješno da ispravi.

Postoje i drugi kodovi poput Rid Solomonovih kodova i turbo kodova koji imaju bolje performanse od Hemingovog koda, ali u okviru ove teze je razmatran Hemingov kod kako bi se na jednostavan način pokazalo da zaštitno kodovanje može da se iskoristi za smanjenje broja grešaka u prenosu.

LITERATURA

[1] Miroslav L. Dukić, „Principi telekomunikacija“, Akademska misao, Beograd, 2008.

[2] http://www.mathworks.com

[3] Miroslav L.Dukić, Goran Marković, Dejan Vujić, „Principi telekomunikacija-Zbornik rešenih problema“, Akademska misao, Beograd, 2008.

[4] Proakis, J.G. „Digital communications“, McGraw Hill, New York, 2000.

[5] Michel C. Jeruchim, Philip Balaban, K.Sam Shanmugan, „Simulation of communication systems, second edition, New York, 2000, e-book

[6] Mathuranathan Viswanatahan, „Simulation of digital communication systems using matlab“, second edition 2013.

[7] Popović M. „Digitalna obrada signala“, Akademska misao, Beograd, 2003.

[8] Ljiljana Milić, Zoran Dobrosavljević, „Uvod u digitalnu obradu signala“, Akademska misao, Beograd, 2009.

A. KOD REALIZOVAN U MATLABU

A.1. 4-QAM MODULACIJA

```
clc, clear all, close all;
 for i = 1:5
 Nb = 3000;                                  % broj informacionih bita
podaci_unipol = randi([0 1],1,Nb);           % generisanje informacionih bita
%%% zastitno kodovanje HAMMING (8,4)
n_hamming = 8;
k_hamming = 4;
AAA = reshape(podaci_unipol, k_hamming,Nb/k_hamming);     % matricni prikaz informacije
AAA = AAA';
% pomocne matrice za odredjivanje zastitnih bita
z1_bit = [];
z2_bit = [];
z3_bit = [];
for ii = 1:Nb/k_hamming
   z1 = mod(AAA(ii,1) + AAA(ii,2) + AAA(ii,4), 2);      % odredjivanje prvog zastitnog bita za svaku kodnu rijec
   z2 = mod(AAA(ii,1) + AAA(ii,3) + AAA(ii,4), 2);      % odredjivanje drugog zastitnog bita za svaku kodnu rijec
   z3 = mod(AAA(ii,2) + AAA(ii,3) + AAA(ii,4), 2);      % odredjivanje treceg zastitnog bita za svaku kodnu rijec
  z1_bit = [z1_bit z1];        % pomocni vektor prvog zastitnog bita
  z2_bit = [z2_bit z2];        % pomocni vektor drugog zastitnog bita
  z3_bit = [z3_bit z3];        % pomocni vektor treceg zastitnog bita
 end
Zast_bit = [z1_bit' z2_bit' z3_bit'];                % pomocna matrica zastitnih bita
hamming_matrica = [AAA Zast_bit];                    % spajanje informacionih i zastitnih bita
```

```
redosled_kolona = [5 6 1 7 2 3 4];
hamming_sedam_cetiri = hamming_matrica(:,redosled_kolona);      % kreiranje kodnih rijeci, biti postavljeni na svoje mjesto
 parnost = mod(sum(hamming_sedam_cetiri'), 2);                  % odredjivanje bita provjere parnosti
hamming_osam_cetiri = [hamming_sedam_cetiri parnost'];  % dodavanje cetvrtog zastitnog bita
[nn, mm] = size(hamming_osam_cetiri);
hamming_linija = reshape(hamming_osam_cetiri', 1, nn*mm);
podaci_bipol = 2*hamming_linija-1;                  % kreiranje bipolarnih impulsa
Br_simb = length(podaci_bipol)/2;                   % broj simbola
podaci_s_p = reshape(podaci_bipol,2,Br_simb);       % serijski u paralelne nizove
Vb = 250000;                                        % bitska brzina
fc = Vb;                                            % noseca ucestanost
T = 1/fc;                                           % period jednog simbola/bita
N_tacaka = 16;                                      % broj tacaka po bitu/simbolu
t = T/N_tacaka:T/N_tacaka:T;                        % vremenska osa jednog bita/simbola
y = [];
y_in = [];
y_qd = [];
for n = 1:Br_simb
    y1 = podaci_s_p(1,n)*cos(2*pi*fc*t+pi/4)*2;
    y2 = podaci_s_p(2,n)*1i*sin(2*pi*fc*t+pi/4)*2;
    y_in = [y_in y1];
    y_qd = [y_qd y2];
    y = [y y1+y2];
end
scatterplot(y',N_tacaka,(N_tacaka/2)-1),title('konstelacioni dijagram na predaji'); % konstelacioni dijagram
tt = T/N_tacaka:T/N_tacaka:(T*length(podaci_bipol))/2;      % vremenska osa kompletnog signala
figure,
subplot(311), plot(tt,real(y_in)),ylabel('real(y(t))'),xlabel('t'),title('realni deo modulisanog signala u vremenu');
subplot(312), plot(tt,imag(y_qd)),ylabel('imag(y(t))'),xlabel('t'),title('imaginarni deo modulisanog signala u vremenu');
```

```
subplot(313),
plot(tt,real(y_in)+imag(y_qd)),ylabel('y(t)'),xlabel('t'),title('modulisani signal u vremenu');
y_noise = awgn(y,-4);
figure,
subplot(211),
plot(tt,real(y_in)+imag(y_qd)),ylabel('y(t)'),xlabel('t'),title('modulisani signal u vremenu na predaji');
subplot(212),
plot(tt,real(y_noise)+imag(y_noise)),ylabel('y(t)'),xlabel('t'),title('modulisani signal u vremenu na prijemu');
scatterplot(y_noise',N_tacaka,(N_tacaka/2)-1),title('konstelacioni dijagram na prijemu');
Podaci_prijem = [];
for n = 1:Br_simb
  z_in = real(y_noise((n-1)*N_tacaka+1:n*N_tacaka)).*cos(2*pi*fc*t+pi/4)*4;
  z_in_int = (mean(z_in));
   if(z_in_int>0)
      Podaci_prijem_in = 1;
   else
      Podaci_prijem_in = 0;
end
 z_qd = imag(y_noise((n-1)*N_tacaka+1:n*N_tacaka)).*sin(2*pi*fc*t+pi/4).*4;
 z_qd_int = (mean(z_qd));
 if(z_qd_int>0)
      Podaci_prijem_qd = 1;
  else
      Podaci_prijem_qd = 0;
end
Podaci_prijem = [Podaci_prijem Podaci_prijem_in Podaci_prijem_qd];
inphase(n) = z_in_int;
qudrature(n) = z_qd_int;
end
%%%%%% zastitno dekodovanje HAMMING (8,4)
podaci_prijem_matrica = reshape(Podaci_prijem, n_hamming, Nb/k_hamming);
```

```
% kreiranje matrice, svaki red predstavlja primljenu kodnu rijec
podaci_prijem_matrica = podaci_prijem_matrica';
%%% pomocni vektori sindroma
s1_bit = [];
s2_bit = [];
s3_bit = [];
s4_bit = [];
for jj = 1:Nb/k_hamming
  %%%% odredjivanje sindroma
   s1 = mod(podaci_prijem_matrica(jj,1) + podaci_prijem_matrica(jj,3) +
podaci_prijem_matrica(jj,5) + podaci_prijem_matrica(jj,7), 2);
   s2 = mod(podaci_prijem_matrica(jj,2) + podaci_prijem_matrica(jj,3) +
podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7), 2);
   s3 = mod(podaci_prijem_matrica(jj,4) + podaci_prijem_matrica(jj,5) +
podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7), 2);
  s4 = mod(podaci_prijem_matrica(jj,1) + podaci_prijem_matrica(jj,2) +
podaci_prijem_matrica(jj,3) + podaci_prijem_matrica(jj,4) + ...
      podaci_prijem_matrica(jj,5) + podaci_prijem_matrica(jj,6) +
podaci_prijem_matrica(jj,7) + podaci_prijem_matrica(jj,8),2);
 s1_bit = [s1_bit s1];
 s2_bit = [s2_bit s2];
 s3_bit = [s3_bit s3];
 s4_bit = [s4_bit s4];
end
Sindrom = [s3_bit' s2_bit' s1_bit'];      % svaki red predstavlja sindrom za
odgovarajucu kodnu rijec
%%%% pomocni vektori A
A1 = [1 0 0 0 0 0 0 0];
A2 = [0 1 0 0 0 0 0 0];
A3 = [0 0 1 0 0 0 0 0];
A4 = [0 0 0 1 0 0 0 0];
A5 = [0 0 0 0 1 0 0 0];
A6 = [0 0 0 0 0 1 0 0];
A7 = [0 0 0 0 0 0 1 0];
A8 = [0 0 0 0 0 0 0 1];
```

```
%%%%% korekcija i detekcija gresaka
ispravljeno = podaci_prijem_matrica;
for kk = 1:Nb/k_hamming
   if Sindrom(kk,:) == [0 0 1];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A1, 2);
   elseif Sindrom(kk,:) == [0 1 0];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A2, 2);
   elseif Sindrom(kk,:) == [0 1 1];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A3, 2);
   elseif Sindrom(kk,:) == [1 0 0];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A4, 2);
   elseif Sindrom(kk,:) == [1 0 1];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A5, 2);
   elseif Sindrom(kk,:) == [1 1 0];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A6, 2);
   elseif Sindrom(kk,:) == [1 1 1];
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A7, 2);
   elseif (Sindrom(kk,:) == [0 0 0]) & (s4_bit(kk) == [1]);
      ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A8, 2);
   end
end
podaci_prijem_dekodovano = reshape(ispravljeno',1,(Nb/k_hamming)*8);
xx = [3 5 6 7 1 2 4 8];
prijem_korisnik_matrica = ispravljeno(:,xx);
prijem_korisnik_matrica = prijem_korisnik_matrica(:,1:4);
prijem_korisnik = reshape(prijem_korisnik_matrica',1,Nb);
[broj_gresaka_hamm(i),BER_hamm(i)] = biterr(hamming_linija,Podaci_prijem);
% mjerenje BER-a i broja bitskih gresaka na prijemu prije zas.dekodovanja
[broj_gresaka_dekodovano(i),BER_dekodovano(i)] = biterr(hamming_linija,podaci_prijem_dekodovano); % mjerenje BER-a i broja bitskih gresaka na prijemu nakon dekodovanja
[broj_gresaka(i),BER(i)] = biterr(podaci_unipol,prijem_korisnik); % mjerenje BER-a i broja bitskih gresaka na prijemu posle zas. Dekodovanja
Rezultati_hamm=[broj_gresaka_hamm'BER_hamm'];Rezultati_dekodovano=
[broj_gresaka_dekodovano',BER_dekodovano'];Rezultati = [broj_gresaka' BER'];
end
```

A.2. 16-*QAM* MODULACIJA

```
clc, clear all, close all;
for i = 1:5
Nb = 3000;                                  % broj informacionih bita (MORA BITI DJELJIV SA CETIRI)
podaci = randi([0 1],1,Nb);                 % generisanje informacionih bita
%%% zastitno kodovanje HAMMING (8,4)
n_hamming = 8;
k_hamming = 4;
AAA = reshape(podaci, k_hamming,Nb/k_hamming);      % matricni prikaz informacije
AAA = AAA';
% pomocne matrice za odredjivanje zastitnih bita
z1_bit = [];
z2_bit = [];
z3_bit = [];
for ii = 1:Nb/k_hamming
    z1 = mod(AAA(ii,1) + AAA(ii,2) + AAA(ii,4), 2);          % odredjivanje prvog zastitnog bita za svaku kodnu rijec
    z2 = mod(AAA(ii,1) + AAA(ii,3) + AAA(ii,4), 2);          % odredjivanje drugog zastitnog bita za svaku kodnu rijec
    z3 = mod(AAA(ii,2) + AAA(ii,3) + AAA(ii,4), 2);          % odredjivanje treceg zastitnog bita za svaku kodnu rijec
    z1_bit = [z1_bit z1];             % pomocni vektor prvog zastitnog bita
    z2_bit = [z2_bit z2];             % pomocni vektor drugog zastitnog bita
    z3_bit = [z3_bit z3];             % pomocni vektor treceg zastitnog bita


end
Zast_bit = [z1_bit' z2_bit' z3_bit'];                       % pomocna matrica zastitnih bita
hamming_matrica = [AAA Zast_bit];                           % spajanje informacionih i zastitnih bita
redosled_kolona = [5 6 1 7 2 3 4];
hamming_sedam_cetiri = hamming_matrica(:,redosled_kolona);          % kreiranje kodnih rijeci, biti postavljeni na svoje mjestoparnost = mod(sum(hamming_sedam_cetiri'), 2);              % odredjivanje bita
```

```
provjere    parnostihamming_osam_cetiri    =    [hamming_sedam_cetiri
parnost'];  % dodavanje cetvrtog zastitnog bita
[nn, mm] = size(hamming_osam_cetiri);
hamming_linija = reshape(hamming_osam_cetiri', 1, nn*mm);
%%%%%% priprema za modulaciju
duzina = length(hamming_linija)/2;                    % priprema za
prelazak iz serijskog u paralelne nizove
% serijski u paralelne nizove
podaci_1_grana = hamming_linija(1:duzina);
podaci_2_grana = hamming_linija(duzina+1:end);
podaci_s_p = [podaci_1_grana; podaci_2_grana];
% odredjivanje nivoa signala I ose
Nivo_in = [];
for n = 1:2:length(podaci_s_p);
    if podaci_s_p(1,n)==0 && podaci_s_p(1,n+1)==0;
        x1=(-3);
    elseif podaci_s_p(1,n)==0 && podaci_s_p(1,n+1)==1;
        x1=(-1);
    elseif podaci_s_p(1,n)==1 && podaci_s_p(1,n+1)==0;
        x1=(1);
    else podaci_s_p(1,n)==1 && podaci_s_p(1,n+1)==1;
        x1=(3);
    end
    Nivo_in = [Nivo_in x1];
end
% odredjivanje nivoa signala Q ose
Nivo_qd = [];
for n = 1:2:length(podaci_s_p);
    if podaci_s_p(2,n)==0 && podaci_s_p(2,n+1)==0;
        x2=(-3);
    elseif podaci_s_p(2,n)==0 && podaci_s_p(2,n+1)==1;
        x2=(-1);
    elseif podaci_s_p(2,n)==1 && podaci_s_p(2,n+1)==0;
        x2=(1);
    else podaci_s_p(2,n)==1 && podaci_s_p(2,n+1)==1;
        x2=(3);
    end
    Nivo_qd = [Nivo_qd x2];
end
```

```
Vb = 250000;                        % bitska brzina
fc = Vb;                            % noseca ucestanost
T = 1/fc;                           % period jednog simbola/bita
N_tacaka = 16;                      % broj tacaka po bitu/simbolu
t = T/N_tacaka:T/N_tacaka:T;        % vremenska osa jednog bita/simbola
% modulisanje signala
y = [];
y_in = [];
y_qd = [];
for n = 1:duzina/2
    y1 = Nivo_in(1,n)*cos(2*pi*fc*t+pi/4);
    y2 = Nivo_qd(1,n)*1i*sin(2*pi*fc*t+pi/4);
    y_in = [y_in y1];
    y_qd = [y_qd y2];
    y = [y y1+y2];
end
scatterplot(y',N_tacaka,(N_tacaka/2)-1),title('konstelacioni dijagram na predaji');
% konstelacioni dijagram na predaji
tt = T/N_tacaka:T/N_tacaka:(T*length(hamming_linija))/4;      % vremenska osa kompletnog signala
figure,
subplot(311), plot(tt,real(y_in)),ylabel('real(y(t))'),xlabel('t'),title('realni deo modulisanog signala u vremenu');
subplot(312), plot(tt,imag(y_qd)),ylabel('imag(y(t))'),xlabel('t'),title('imaginarni deo modulisanog signala u vremenu');
subplot(313),
plot(tt,real(y_in)+imag(y_qd)),ylabel('y(t)'),xlabel('t'),title('modulisani signal u vremenu');
% prolazak signala kroz kanal (dodavanje suma)
y_noise = awgn(y,0);
figure,subplot(211),
plot(tt,real(y_in)+imag(y_qd)),ylabel('y(t)'),xlabel('t'),title('modulisani signal u vremenu na predaji');
subplot(212),
plot(tt,real(y_noise)+imag(y_noise)),ylabel('y(t)'),xlabel('t'),title('modulisani signal u vremenu na prijemu');
```

```
catterplot(y_noise',N_tacaka,(N_tacaka/2)-1),title('konstelacioni dijagram na prijemu');

% demodulacija i odlucivac
 Podaci_prijem_in_sve = [];
Podaci_prijem_qd_sve = [];
for n = 1:duzina/2
    z_in = real(y_noise((n-1)*N_tacaka+1:n*N_tacaka)).*cos(2*pi*fc*t+pi/4)*2;
    z_in_int = (mean(z_in));
  if(z_in_int>2)
       Podaci_prijem_in = [1 1];
   elseif (2 >= z_in_int) &&  (z_in_int > 0)
       Podaci_prijem_in = [1 0];
   elseif (0 >= z_in_int) &&  (z_in_int > (-2))
       Podaci_prijem_in = [0 1];
   else
       Podaci_prijem_in = [0 0];
   End
Podaci_prijem_in_sve = [Podaci_prijem_in_sve Podaci_prijem_in];
end
for n = 1:duzina/2
    z_qd = imag(y_noise((n-1)*N_tacaka+1:n*N_tacaka)).*sin(2*pi*fc*t+pi/4)*2;
    z_qd_int = (mean(z_qd));
  if(z_qd_int > 2)
       Podaci_prijem_qd = [1 1];
  elseif (2 >= z_qd_int) &&  (z_qd_int > 0)
       Podaci_prijem_qd = [1 0];
  elseif (0 >= z_qd_int) &&  (z_qd_int > (-2))
       Podaci_prijem_qd = [0 1];
   else
       Podaci_prijem_qd = [0 0];
   end
 Podaci_prijem_qd_sve = [Podaci_prijem_qd_sve Podaci_prijem_qd];
 end
```

```
Podaci_prijem = [Podaci_prijem_in_sve Podaci_prijem_qd_sve];
%%%%%% zastitno dekodovanje HAMMING (8,4)podaci_prijem_matrica =
reshape(Podaci_prijem, n_hamming, Nb/k_hamming);        % kreiranje
matrice, svaki red predstavlja primljenu kodnu rijec
podaci_prijem_matrica = podaci_prijem_matrica';
%%% pomocni vektori sindroma
s1_bit = [];
s2_bit = [];
s3_bit = [];
s4_bit = [];
for jj = 1:Nb/k_hamming
%%%% odredjivanje sindroma
    s1=mod(podaci_prijem_matrica(jj,1) + podaci_prijem_matrica(jj,3)
+ podaci_prijem_matrica(jj,5) + podaci_prijem_matrica(jj,7), 2);
    s2=mod(podaci_prijem_matrica(jj,2) + podaci_prijem_matrica(jj,3)
+ podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7), 2);
    s3=mod(podaci_prijem_matrica(jj,4) + podaci_prijem_matrica(jj,5)
+ podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7), 2);


    s4=mod(podaci_prijem_matrica(jj,1) + podaci_prijem_matrica(jj,2)
+ podaci_prijem_matrica(jj,3) + podaci_prijem_matrica(jj,4) + ...
        podaci_prijem_matrica(jj,5) + podaci_prijem_matrica(jj,6) +
podaci_prijem_matrica(jj,7) + podaci_prijem_matrica(jj,8),2);
    s1_bit = [s1_bit s1];
    s2_bit = [s2_bit s2];
    s3_bit = [s3_bit s3];
    s4_bit = [s4_bit s4];
end
Sindrom = [s3_bit' s2_bit' s1_bit'];      % svaki red predstavlja
sindrom za odgovarajucu kodnu rijec
%%%% pomocni vektori A
A1 = [1 0 0 0 0 0 0 0];
A2 = [0 1 0 0 0 0 0 0];
A3 = [0 0 1 0 0 0 0 0];
A4 = [0 0 0 1 0 0 0 0];
A5 = [0 0 0 0 1 0 0 0];
A6 = [0 0 0 0 0 1 0 0];
A7 = [0 0 0 0 0 0 1 0];
A8 = [0 0 0 0 0 0 0 1];
%%%%% korekcija i detekcija gresaka
ispravljeno = podaci_prijem_matrica;
```

```
for kk = 1:Nb/k_hamming
    if Sindrom(kk,:) == [0 0 1];
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A1, 2);
    elseif Sindrom(kk,:) == [0 1 0];
ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A2, 2);
    elseif Sindrom(kk,:) == [0 1 1];
ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A3, 2);
    elseif Sindrom(kk,:) == [1 0 0];
ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A4, 2);
    elseif Sindrom(kk,:) == [1 0 1];
ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A5, 2);
    elseif Sindrom(kk,:) == [1 1 0];
ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A6, 2);
    elseif Sindrom(kk,:) == [1 1 1];
ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A7, 2);
    elseif (Sindrom(kk,:) == [0 0 0]) & (s4_bit(kk) == [1]);
ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A8, 2);
    end
end
podaci_prijem_dekodovano = reshape(ispravljeno',1,(Nb/k_hamming)*8);
xx = [3 5 6 7 1 2 4 8];
prijem_korisnik_matrica = ispravljeno(:,xx);
prijem_korisnik_matrica = prijem_korisnik_matrica(:,1:4);          %
postavljane kolona na svoje mjesto i uzimanje informacionih bita
prijem_korisnik = reshape(prijem_korisnik_matrica',1,Nb);
[broj_gresaka_hamm(i),BER_hamm(i)]                                 =
biterr(hamming_linija,Podaci_prijem);              % mjerenje BER-a i
broja bitskih gresaka na prijemu prije zas.dekodovanja
[broj_gresaka_dekodovano(i),BER_dekodovano(i)]                     =
biterr(hamming_linija,podaci_prijem_dekodovano);  % mjerenje BER-a i
broja bitskih gresaka na prijemu nakon dekodovanja
[broj_gresaka(i),BER(i)]=biterr(podaci,prijem_korisnik);
% mjerenje BER-a i broja bitskih gresaka na prijemu posle zas.
dekodovanja
Rezultati_hamm = [broj_gresaka_hamm' BER_hamm'];
Rezultati_dekodovano = [broj_gresaka_dekodovano',BER_dekodovano'];
Rezultati = [broj_gresaka' BER'];
 End
```

A.3.64-*QAM* MODULACIJA

```
clc, clear all, close all;
for i = 1:5
Nb = 3000;                          % broj informacionih bita
podaci = randi([0 1],1,Nb);              % generisanje informacionih bita
%%% zastitno kodovanje HAMMING (8,4)
n_hamming = 8;
k_hamming = 4;
AAA = reshape(podaci, k_hamming,Nb/k_hamming);        % matricni prikaz informacije
AAA = AAA';
% pomocne matrice za odredjivanje zastitnih bita
z1_bit = [];
z2_bit = [];
z3_bit = [];
for ii = 1:Nb/k_hamming
   z1 = mod(AAA(ii,1) + AAA(ii,2) + AAA(ii,4), 2);      % odredjivanje prvog zastitnog bita za svaku kodnu rijec
   z2 = mod(AAA(ii,1) + AAA(ii,3) + AAA(ii,4), 2);      % odredjivanje drugog zastitnog bita za svaku kodnu rijec
   z3 = mod(AAA(ii,2) + AAA(ii,3) + AAA(ii,4), 2);      % odredjivanje treceg zastitnog bita za svaku kodnu rijec
   z1_bit = [z1_bit z1];         % pomocni vektor prvog zastitnog bita
   z2_bit = [z2_bit z2];         % pomocni vektor drugog zastitnog bita
  z3_bit = [z3_bit z3];         % pomocni vektor treceg zastitnog bita
end
Zast_bit = [z1_bit' z2_bit' z3_bit'];              % pomocna matrica zastitnih bita
hamming_matrica = [AAA Zast_bit];                  % spajanje informacionih i zastitnih bita
redosled_kolona = [5 6 1 7 2 3 4];
```

```
hamming_sedam_cetiri = hamming_matrica(:,redosled_kolona);      % kreiranje kodnih rijeci, biti postavljeni na svoje mjesto

parnost = mod(sum(hamming_sedam_cetiri'), 2);          % odredjivanje bita provjere parnostihamming_osam_cetiri = [hamming_sedam_cetiri parnost']; % dodavanje cetvrtog zastitnog bita

[nn, mm] = size(hamming_osam_cetiri);

hamming_linija = reshape(hamming_osam_cetiri', 1, nn*mm);

duzina = length(hamming_linija)/2;                    % priprema za prelazak iz serijskog u paralelne nizove

% serijski u paralelne nizove

podaci_1_grana = hamming_linija(1:duzina);

podaci_2_grana = hamming_linija(duzina+1:end);

podaci_s_p = [podaci_1_grana; podaci_2_grana];

% odredjivanje nivoa signala I ose

Nivo_in = [];

for n = 1:3:length(podaci_s_p);

    if     podaci_s_p(1,n)==0     &&     podaci_s_p(1,n+1)==0     && podaci_s_p(1,n+2)==0;

        x1=(-3.5);

    elseif    podaci_s_p(1,n)==0    &&    podaci_s_p(1,n+1)==0    && podaci_s_p(1,n+2)==1;

        x1=(-2.5);

    elseif    podaci_s_p(1,n)==0    &&    podaci_s_p(1,n+1)==1    && podaci_s_p(1,n+2)==0;

        x1=(-1.5);

    elseif    podaci_s_p(1,n)==0    &&    podaci_s_p(1,n+1)==1    && podaci_s_p(1,n+2)==1;

        x1=(-0.5);

    elseif    podaci_s_p(1,n)==1    &&    podaci_s_p(1,n+1)==0    && podaci_s_p(1,n+2)==0;

        x1=0.5;

    elseif    podaci_s_p(1,n)==1    &&    podaci_s_p(1,n+1)==0    && podaci_s_p(1,n+2)==1;

        x1=1.5;

    elseif    podaci_s_p(1,n)==1    &&    podaci_s_p(1,n+1)==1    && podaci_s_p(1,n+2)==0;

        x1=2.5;

    else     podaci_s_p(1,n)==1     &&     podaci_s_p(1,n+1)==1     && podaci_s_p(1,n+2)==1;

        x1=3.5;

    end

    Nivo_in = [Nivo_in x1];
```

```
end

% odredjivanje nivoa signala Q ose
Nivo_qd = [];
for n = 1:3:length(podaci_s_p);
    if      podaci_s_p(2,n)==0      &&      podaci_s_p(2,n+1)==0      &&
podaci_s_p(2,n+2)==0;
        x2=(-3.5);
    elseif     podaci_s_p(2,n)==0     &&     podaci_s_p(2,n+1)==0     &&
podaci_s_p(2,n+2)==1;
        x2=(-2.5);
    elseif     podaci_s_p(2,n)==0     &&     podaci_s_p(2,n+1)==1     &&
podaci_s_p(2,n+2)==0;
        x2=(-1.5);
    elseif     podaci_s_p(2,n)==0     &&     podaci_s_p(2,n+1)==1     &&
podaci_s_p(2,n+2)==1;
        x2=(-0.5);
    elseif     podaci_s_p(2,n)==1     &&     podaci_s_p(2,n+1)==0     &&
podaci_s_p(2,n+2)==0;
        x2=0.5;
    elseif     podaci_s_p(2,n)==1     &&     podaci_s_p(2,n+1)==0     &&
podaci_s_p(2,n+2)==1;
        x2=1.5;
    elseif     podaci_s_p(2,n)==1     &&     podaci_s_p(2,n+1)==1     &&
podaci_s_p(2,n+2)==0;
        x2=2.5;
    else     podaci_s_p(2,n)==1      &&      podaci_s_p(2,n+1)==1      &&
podaci_s_p(2,n+2)==1;
        x2=3.5;
    end
    Nivo_qd = [Nivo_qd x2];
end

Vb = 250000;                                        % bitska brzina
fc = Vb;                                            % noseca ucestanost
T = 1/fc;                                               % period jednog
simbola/bita
N_tacaka = 16;                                        % broj tacaka po
bitu/simbolu
t = T/N_tacaka:T/N_tacaka:T;                           % vremenska osa
jednog bita/simbola
% modulisanje signala
```

```
y = [];
y_in = [];
y_qd = [];
for n = 1:duzina/3
    y1 = Nivo_in(1,n)*cos(2*pi*fc*t+pi/4);
    y2 = Nivo_qd(1,n)*1i*sin(2*pi*fc*t+pi/4);
    y_in = [y_in y1];
    y_qd = [y_qd y2];
    y = [y y1+y2];
end
scatterplot(y',N_tacaka,(N_tacaka/2)-1),title('konstelacioni
dijagram na predaji');                  % konstelacioni dijagram
tt  =  T/N_tacaka:T/N_tacaka:(T*length(hamming_linija))/6;            %
vremenska osa kompletnog signala
figure,
subplot(311),
plot(tt,real(y_in)),ylabel('real(y(t))'),xlabel('t'),title('realni
deo modulisanog signala u vremenu');
subplot(312),
plot(tt,imag(y_qd)),ylabel('imag(y(t))'),xlabel('t'),title('imaginar
ni deo modulisanog signala u vremenu');
subplot(313),
plot(tt,real(y_in)+imag(y_qd)),ylabel('y(t)'),xlabel('t'),title('mod
ulisani signal u vremenu');
% prolazak signala kroz kanal (dodavanje suma)
y_noise = awgn(y,6);
figure,
subplot(211),
plot(tt,real(y_in)+imag(y_qd)),ylabel('y(t)'),xlabel('t'),title('mod
ulisani signal u vremenu na predaji');
subplot(212),
plot(tt,real(y_noise)+imag(y_noise)),ylabel('y(t)'),xlabel('t'),titl
e('modulisani signal u vremenu na prijemu');
scatterplot(y_noise',N_tacaka,(N_tacaka/2)-1),title('konstelacioni
dijagram na prijemu');
% demodulacija i odlucivac
Podaci_prijem_in_sve = [];
Podaci_prijem_qd_sve = [];
for n = 1:duzina/3
    z_in                    =                    real(y_noise((n-
1)*N_tacaka+1:n*N_tacaka)).*cos(2*pi*fc*t+pi/4)*2;
    z_in_int = (mean(z_in));
if(z_in_int>3)
```

```
        Podaci_prijem_in = [1 1 1];
    elseif (3 >= z_in_int) &&  (z_in_int > 2)
        Podaci_prijem_in = [1 1 0];
    elseif (2 >= z_in_int) &&  (z_in_int > 1)
        Podaci_prijem_in = [1 0 1];
    elseif (1 >= z_in_int) &&  (z_in_int > 0)
        Podaci_prijem_in = [1 0 0];
    elseif (0 >= z_in_int) &&  (z_in_int > (-1))
        Podaci_prijem_in = [0 1 1];
    elseif ((-1) >= z_in_int) &&  (z_in_int > (-2))
        Podaci_prijem_in = [0 1 0];
    elseif ((-2) >= z_in_int) &&  (z_in_int > (-3))
        Podaci_prijem_in = [0 0 1]
     else
        Podaci_prijem_in = [0 0 0];
    end
    Podaci_prijem_in_sve = [Podaci_prijem_in_sve Podaci_prijem_in];
end
for n=1:duzina/3
     z_qd=imag(y_noise((n-1)*N_tacaka+1:n*N_tacaka))
     .*sin(2*pi*fc*t+pi/4)*2 ;
    z_qd_int = (mean(z_qd));
    if(z_qd_int>3)
        Podaci_prijem_qd = [1 1 1];
    elseif (3 >= z_qd_int) &&  (z_qd_int > 2)
        Podaci_prijem_qd = [1 1 0];
    elseif (2 >= z_qd_int) &&  (z_qd_int > 1)
        Podaci_prijem_qd = [1 0 1];
    elseif (1 >= z_qd_int) &&  (z_qd_int > 0)
        Podaci_prijem_qd = [1 0 0];
    elseif (0 >= z_qd_int) &&  (z_qd_int > (-1))
        Podaci_prijem_qd = [0 1 1];
    elseif ((-1) >= z_qd_int) &&  (z_qd_int > (-2))
        Podaci_prijem_qd = [0 1 0];
    elseif ((-2) >= z_qd_int) &&  (z_qd_int > (-3))
        Podaci_prijem_qd = [0 0 1];
    else
        Podaci_prijem_qd = [0 0 0];
```

```
    end


    Podaci_prijem_qd_sve = [Podaci_prijem_qd_sve Podaci_prijem_qd];
end
Podaci_prijem = [Podaci_prijem_in_sve Podaci_prijem_qd_sve];
%%%%%% zastitno dekodovanje HAMMING (8,4)
podaci_prijem_matrica=reshape(Podaci_prijem,n_hamming,Nb/k_hamming);
% kreiranje matrice, svaki red predstavlja primljenu kodnu rijec
podaci_prijem_matrica = podaci_prijem_matrica';%%% pomocni vektori
sindroma
s1_bit = [];
s2_bit = [];
s3_bit = [];
s4_bit = [];
for jj = 1:Nb/k_hamming
%%%% odredjivanje sindroma
    s1=mod(podaci_prijem_matrica(jj,1) + podaci_prijem_matrica(jj,3)
+ podaci_prijem_matrica(jj,5) + podaci_prijem_matrica(jj,7), 2);
    s2=mod(podaci_prijem_matrica(jj,2) + podaci_prijem_matrica(jj,3)
+ podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7), 2);
    s3=mod(podaci_prijem_matrica(jj,4) + podaci_prijem_matrica(jj,5)
+ podaci_prijem_matrica(jj,6) + podaci_prijem_matrica(jj,7), 2);
    s4=mod(podaci_prijem_matrica(jj,1) + podaci_prijem_matrica(jj,2)
+ podaci_prijem_matrica(jj,3) + podaci_prijem_matrica(jj,4) + ...
 podaci_prijem_matrica(jj,5)+podaci_prijem_matrica(jj,6)
+podaci_prijem_matrica(jj,7) + podaci_prijem_matrica(jj,8),2);
    s1_bit = [s1_bit s1];
    s2_bit = [s2_bit s2];
    s3_bit = [s3_bit s3];
    s4_bit = [s4_bit s4];
end
Sindrom = [s3_bit' s2_bit' s1_bit'];      % svaki red predstavlja
sindrom za odgovarajucu kodnu rijec
%%%% pomocni vektori A
A1 = [1 0 0 0 0 0 0 0];
A2 = [0 1 0 0 0 0 0 0];
A3 = [0 0 1 0 0 0 0 0];
A4 = [0 0 0 1 0 0 0 0];
A5 = [0 0 0 0 1 0 0 0];
A6 = [0 0 0 0 0 1 0 0];
A7 = [0 0 0 0 0 0 1 0];
A8 = [0 0 0 0 0 0 0 1];
```

```
%%%%% korekcija i detekcija gresaka
ispravljeno = podaci_prijem_matrica;
for kk = 1:Nb/k_hamming

    if Sindrom(kk,:) == [0 0 1];
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A1, 2);
    elseif Sindrom(kk,:) == [0 1 0];
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A2, 2);
    elseif Sindrom(kk,:) == [0 1 1];
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A3, 2);
    elseif Sindrom(kk,:) == [1 0 0];
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A4, 2);
    elseif Sindrom(kk,:) == [1 0 1];
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A5, 2);
    elseif Sindrom(kk,:) == [1 1 0];
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A6, 2);
    elseif Sindrom(kk,:) == [1 1 1];
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A7, 2);
    elseif (Sindrom(kk,:) == [0 0 0]) & (s4_bit(kk) == [1]);
       ispravljeno(kk,:) = mod(podaci_prijem_matrica(kk,:) + A8, 2);
    end
end
podaci_prijem_dekodovano = reshape(ispravljeno',1,(Nb/k_hamming)*8);
xx = [3 5 6 7 1 2 4 8];
prijem_korisnik_matrica = ispravljeno(:,xx);
prijem_korisnik_matrica = prijem_korisnik_matrica(:,1:4);          %
postavljane kolona na svoje mjesto i uzimanje informacionih bita
prijem_korisnik = reshape(prijem_korisnik_matrica',1,Nb);
[broj_gresaka_hamm(i),BER_hamm(i)]                                 =
biterr(hamming_linija,Podaci_prijem);              % mjerenje BER-a i
broja bitskih gresaka na prijemu prije zas.dekodovanja
[broj_gresaka_dekodovano(i),BER_dekodovano(i)]                     =
biterr(hamming_linija,podaci_prijem_dekodovano);  % mjerenje BER-a i
broja bitskih gresaka na prijemu nakon dekodovanja
[broj_gresaka(i),BER(i)]=biterr(podaci,prijem_korisnik);
% mjerenje BER-a i broja bitskih gresaka na prijemu posle zas.
dekodovanja
Rezultati_hamm = [broj_gresaka_hamm' BER_hamm'];
Rezultati_dekodovano = [broj_gresaka_dekodovano',BER_dekodovano'];
Rezultati = [broj_gresaka' BER'];
end
```