Gheorghe M.Panaitescu

Aplicații la disciplina

TRANSMITEREA ŞI CODAREA INFORMAȚIEI

Universitatea Petrol-Gaze Ploiesti
2013

CUVÂNT ÎNAINTE

Volumul acesta este o colectie de aplicatii dezvoltate/propuse în sprijinul întelegerii mai adânci a disciplinei **Transmiterea si codarea informatiei** predată anului II de la specializarea Electronică aplicată.

În parte, aplicatiile sunt produse de autor. Partea majoritară însă este preluată din literatură. Au fost consultate câteva sute de probleme destinate studentilor de pe alte meridiane, unele au fost retinute si au fost traduse si adaptate pentru studentii pe care îi instruiesc.

Solutiile la enunturile de împrumut – un împrumut făcut de cele mai multe ori cu acordul autorilor de origine – sunt si ele în mare parte împrumutate. Unele îmi apartin sau le-am adaptat. Interventia mea în rezolvări este de la caz la caz în proportii diferite si de aceea greu de cuantificat. Dar acesta este un fapt cu totul secundar.

Unele aplicatii se pot trata/rezolva pe cale exclusiv analitică. Altele necesită recursul la calculator. De aceea sunt propuse pe alocuri unele secvente sau chiar programe/scripturi Matlab pentru ca studentii să nu risipească prea mult timp cu elaborarea de programe proprii, ci să se concentreze mai curând asupra problemei si asupra semnificatiei solutiilor obtinute. Desigur, nu este interzis, ci este chiar recomandat ca secventele de program din această lucrare să fie ameliorate sau chiar înlocuite cu altele noi, mai performante.

Aplicatiile sub formă de probleme au uzual solutii complete. Unele au solutii numai schitate. Căile către solutii pot fi desigur si altele, diferite de cele propuse aici. Studentii au sansa de a rezolva prin forte proprii, în variante proprii, probleme în domeniul acesta atât de cuprinzător al transmiterii si codării informatiei.

Suportul teoretic este prezent aici în formă condensată. Teoria *in extenso* poate fi găsită în varii surse. Una din ele este postată pe site-ul catedrei Automatică si calculatoare: http://ac.upg-ploiesti.ro/gpanaitescu/tci.pdf.

Asa cum am afirmat, lucrarea este dedicată studentilor de anul II de la Electronică. Este însă utilă si studentilor de la specializările Automatică si Calculatoare. Sper că si ei, ca si altii o vor găsi utilă.

Autorul

C U P R I N S U L

SEMNALE 121

Lucrarea 9

Tema 1: Serii Fourier

Tema 2: Compensarea fenomenului Gibbs

Lucrarea 10

Tema 1: Eşantionarea semnalelor; reconstituirea din eşantioane

APLICATII DIN TEORIA PROBABILITĂTILOR

Problema 1.

Se alege la întâmplare o lună a anului. Apoi, tot la întâmplare, se alege o zi din acea lună (se admite că anul nu este bisect).

a. Descrieti toate rezultatele (lună, zi) care formează *spatiul probelor* pentru acest experiment aleator.
b. Care este probabilitatea ca luna să fie de 31 de zile?
c. Care este probabilitatea ca ziua aleasă să fie între a zecea (inclusiv) si a douăzecea (inclusiv)?
d. Care este răspunsul la punctul c. dacă anul este bisect?

Solutie.

a. Se enumeră datele calendaristice sub formă de perechi lună-zi (l, z).
b. Prin raportarea numărului de luni de 31 de zile (7) la numărul de luni ale unui an, rezultă 7/12
c. Formula probabilitătii totale conduce la rezultat:
   $\Pr(10 \le z \le 20) = (11/28)(1/12) + (11/30)(4/12) + (11/31)(7/12)$
   Primul termen este pentru februarie, următorul pentru lunile de 30 de zile, ultimul pentru lunile de 31 de zile.
d. Se modifică contributia lunii februarie: în loc de $(11/28)(1/12)$ se pune $(11/29)(1/12)$.

Problema 2.

Fie A, B evenimente produse de un acelasi experiment aleator. Dacă probabilitatea ca cel putin unul din cele două evenimente să se producă este 0,7 si dacă probabilitatea ca cel putin unul din cele două evenimente să nu se producă este 0,6, calculati probabilitatea ca exact unul dintre cele două evenimente să se producă.

Solutie. Este de calculat

$$\Pr(A \Delta B) = \Pr[(A - B) \cup (B - A)] = \Pr(A) + \Pr(B) - 2\Pr(A \cap B)$$

pentru diferenta simetrică $A \Delta B$.

Se scriu relatiile

$$\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B) = 0{,}7$$
$$\Pr(\overline{A} \cup \overline{B}) = \Pr(\overline{A}) + \Pr(\overline{B}) - \Pr(\overline{A} \cap \overline{B}) = 0{,}6$$

Prin adunare se obtine

$$\Pr(A) + \Pr(\overline{A}) + \Pr(B) + \Pr(\overline{B}) - \Pr(A \cap B) - \Pr(\overline{A} \cap \overline{B}) = 1{,}3$$

ceea ce echivalează cu

$$1 + 1 - \Pr(A \cap B) - \Pr(\overline{A \cup B}) = 1{,}3 \quad \text{(de Morgan)}$$

si apoi

$$1 + 1 - \Pr(A \cap B) - (1 - \Pr(A \cup B)) = 1{,}3 \quad \text{(probabilitatea contrarului)}$$

si mai departe

$$\Pr(A \cap B) = 2 - (1 - 0{,}7) - 1{,}3 = 0{,}4$$

Revenind la prima formulă (si la a doua) se obtine

$$\Pr(A \Delta B) = \Pr(A) + \Pr(B) - \Pr(A \cap B) - \Pr(A \cap B) =$$
$$= \Pr(A \cup B) - \Pr(A \cap B) = 0{,}7 - 0{,}4 = 0{,}3$$

Problema 3.

Trei evenimente A, B, C asociate cu un anumit experiment aleator satisfac relatiile următoare:

- $P(A) = 0{,}25$; $P(B) = 0{,}2$; $P(C) = 0{,}25$
- $P(A \cap B) = 0{,}1$; $P(A \cap B \cap C) = 0{,}05$; $P(A \cap C) = 2P(B \cap C)$
- Probabilitatea ca cel putin două din evenimentele A, B, C să se producă este 0,3

a. Calculati probabilitatea ca nici unul dintre cele trei evenimente să nu se producă.
b. Calculati probabilitatea ca să se producă exact unul dintre cele trei evenimente

Solutie. Propozitia a treia a enuntului spune că

$$P[(A \cap B) \cup (A \cap C) \cup (B \cap C)] = 0{,}3$$

Dezvoltată, această relatie conduce la

$$P(A \cap B) + P(A \cap C) + P(B \cap C) - 2P(A \cap B \cap C) = 0{,}3 \quad \text{(v.figura)}$$

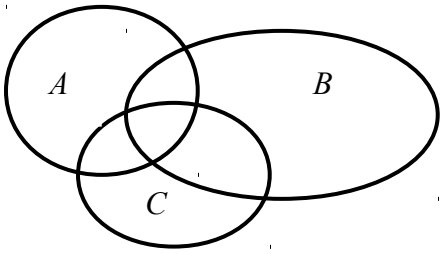


Fig.1. O posibilitate de a judeca cu arii în loc de probabilităti

Prin înlocuirea datelor numerice se obtine

$$P(A \cap C) + P(B \cap C) = 0{,}3 - 0{,}1 + 2.0{,}05 = 0{,}3$$

care alături de

$$P(A \cap C) = 2P(B \cap C)$$

permite evaluarea lui $P(A \cap C) = 0{,}2$ si a lui $P(B \cap C) = 0{,}1$.

Acum se poate trece la evaluările cerute de problemă.

a. Trebuie calculată probabilitatea

$$P(\overline{A} \cap \overline{B} \cap \overline{C}) = P(\overline{A \cup B \cup C}) = 1 - P(A \cup B \cup C)$$

Dar probabilitatea

$$P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B) - P(A \cap C) - P(B \cap C) + \\ + P(A \cap B \cap C)$$

este deplin calculabilă din datele problemei.

b. Trebuie calculată probabilitatea

$$P[(A - B \cup C) \cup (B - A \cup C) \cup (C - A \cup B)]$$

pentru o reuniune de evenimente două câte două mutual incompatibile. Rezultatul este suma probabilităților celor trei evenimente. Se evaluează ca exemplu una din ele

$$P(A - B \cup C) = P[A \cap (\overline{B \cup C})] = \\ = P[A \cap (\overline{B} \cap \overline{C})] = P(A \cap \overline{B} \cap \overline{C}) = P(\overline{A \cup B \cup C})$$

Calculul probabilității de mai sus trece prin relatia cunoscută (si de la punctul a.)

$$P(\overline{A} \cup B \cup C) = P(\overline{A}) + P(B) + P(C) - P(\overline{A} \cap B) - P(\overline{A} \cap C) - P(B \cap C) \\ + \\ + P(\overline{A} \cap B \cap C)$$

Dar

$$P(\overline{A} \cap B) + P(A \cap B) = P(B)$$

din motive de incompatibilitate mutuală a celor două evenimente $\overline{A} \cap B$, $A \cap B$ si pentru că $(\overline{A} \cap B) \cup (A \cap B) = B$. În relatia ultimă există un singur termen necunoscut: $P(\overline{A} \cap B)$.

Analog se evaluează $P(\overline{A} \cap C)$ si $P(\overline{A} \cap B \cap C)$ care, ca si $P(\overline{A} \cap B)$ se înlocuiesc în fromula pentru $P(\overline{A} \cup B \cup C)$.

Rezultă imediat $P(\overline{A \cup B \cup C})$.

Asemănător se evaluează si ceilalti termeni din probabilitatea cerută.

Problema 4.

Se aruncã douã zaruri, unul corect, altul incorect. Cel incorect are probabilitãtile fetelor cu 1, 2, 3, 4, 5, 6 puncte nu egale ci $P(1) = P(2) = P(3) = 2P(4) = 2P(5) = 2P(6)$. Fie X variabila aleatoare care ia valorile de pe zarul corect si Y variabila aleatoare care ia valori conform zarului incorect.

a. Calculati probabilitãtile asociate valorilor variabilei aleatoare Y
b. Calculati valorile medii si dispersiile celor douã variabile aleatoare X si Y
c. Fie Z variabila aleatoare $Z = X - Y$. Valorile posibile ale lui Z sunt 0, ±1, ±2, ±3, ±4, ±5. Evaluati probabilitãtile $P(Z = 0)$, $P(Z = \pm 1)$, $P(Z = \pm 2)$, $P(Z = \pm 3)$, $P(Z = \pm 4)$, $P(Z = \pm 5)$. Faceti o diagramã $P(Z = z)$ cu z în abscisã, pentru $z = 0, \pm 1, \pm 2, \pm 3, \pm 4, \pm 5$.
d. Fie S o secventã de 36 de perechi (i, j), cu $i = 1, 2, 3, 4, 5, 6$ valori ale variabilei aleatoare X, cu j = 1, 2, 3, 4, 5, 6 valori ale variabilei aleatoare Y.

În altă exprimare $X(i, j) = i$, $Y(i, j) = j$ pentru oricare din perechile $(i, j) \in S$. Fie evenimentul $Z = 4$. Calculati probabilitatea ca evenimentul acesta să nu apară nici măcar o dată într-o asemenea secventă.

Solutie.

a. Mai întâi, dacă fetele zarului corect sunt echiprobabile, toate fetele având aceeasi probabilitate de 1/6, la zarul incorect se rezolvă ecuatia $2x + 2x + 2x + x + x + x = 1$ din care rezultă probabilitătile pentru fiecare fată: 2/9, 2/9, 2/9, 1/9, 1/9, 1/9, respectiv pentru $Y = 1, 2, 3, 4, 5, 6$.
b. Se aplică fromulele binecunoscute.
c. Probabilitătile cerute se evaluează prin inventarierea efectivă a cazurilor. De pildă, $Z = \pm 4$ se obtine în cazul perechilor (1, 5'), (2, 6') (5, 1') si (6, 2') (cifrele fără accent sunt obtinute pe zarul corect, cele cu accent pe zarul incorect). Probabilitătile perechilor sunt respectiv (1/6)(1/9), (1/6)(1/9), (1/6)(2/9), (1/6)(2/9) ca produs de probabilităti ale unor evenimente independente. Probabilitatea $P(Z = \pm 4)$ este suma acestor probabilităti (perechile enumerate sunt mutual incompatibile), adică 1/9. La fel se calculează si alte probabilităti cerute prin enunt.
d. Probabilitatea evenimentului $Z = 4$ este 4/54. Probabilitatea evenimentului contrar, $Z \neq 4$ este 50/54. Cele 36 de aruncări succesive sunt independente, evenimentul lipsei totale a evenimentului $Z = 4$ este acelasi lucru cu repetarea (intersectie) de 36 de ori a evenimentului $Z \neq 4$. Probabilitatea acestui eveniment este $(50/54)^{36} \approx 0{,}0626$, o probabilitate destul de mică.

Problema 5.

Fie o secventă de date $\{x_1, x_2, \ldots, x_n\}$ în care fiecare x_i provine din multimea {1, 2, 3, 4}. Din aceste date se poate construi un asa-numit model predictiv care constă în 16 probabilităti conditionate $p(j/i)$, $i, j = 1, 2, 3, 4$ calculate în modul descris mai jos. Pentru fiecare pereche (i, j) fie $N(i)$ numărul de întregi $1 \leq m \leq n$ pentru care $x_m = i$, si fie $N(i, j)$ numărul de întregi $1 \leq m \leq n - 1$ astfel încât $(x_m, x_{m+1}) = (i, j)$. Probabilitatea conditionată căutată este estimată prin

$$p(j / i) = \frac{N(i, j)}{N(i)}$$

a. Fie x o secventă de date pseudoaleatoare de lungime 1000 generată de următorul script Matlab:

```
clear
nk=zeros(1,4);nij=zeros(4,4);samp1=ceil(4*rand); % initializare
for k=1:1000
    samp=ceil(4*rand);
    nk(samp)=nk(samp)+1;
    if k>1
        nij(samp1,samp)=nij(samp1,samp)+1;
        samp1=samp;
    end
end % generare de numere aleatoare uniform pe multimea {1,2,3,4}
```

```
nk % frecvente absolute
nij % frecvebte absolute conditionate
nk=nk/sum(nk) % frecvente relative
s=sum(nij');
for m=1:4
    nij(m,:)=nij(m,:)/s(m);
end
nij % frevente relative conditionate
```

Utilizati scriptul pentru a calcula modelul predictiv $p(j|i)$ pentru secventa x. Exprimati răspunsul ca o matrice de probabilități conditionate 4×4, cu suma pe fiecare linie egală cu unitatea.

b. Pentru modelul predictiv $p(j|i)$ stabilit la punctul anterior, care este cel mai probabil j dacă $i = 1$? Care este cel mai probabil j dacă $i = 2$? Care este cel mai probabil j dacă $i = 3$? Care este cel mai probabil j dacă $i = 4$? (*Indicatie*: Se caută cea mai mare valoare pentru fiecare linie a matricei 4×4 stabilită la punctul a.).

c. Fie acum $x = (x_1, x_2, \ldots, x_{1000})$ secventa generată ca la punctul a. Se admite că modelul predictiv este utilizat în modul descris imediat, pentru a ajunge la o secventă predictivă pentru termenii din x, $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{1000})$.

Se ia prima valoare din secventa predictivă ca fiind $\hat{x}_1 = x_1$. Pentru $1 < m \leq 1000$, se iau predictiile $\hat{x}_m$ pentru x_m ca o valoare $j \in \{1, 2, 3, 4\}$ pentru care probabilitatea $p(j|x_{m-1})$ este cea mai mare (conform punctului b.). Utilizati un script Matlab pentru a genera secventa $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{1000})$ potrivit aceastei reguli de predictie simple. Calculati probabilitatea empirică a predictiei eronate, cu alte cuvinte, numărati pentru câti de m, $1 < m \leq 1000$ $\hat{x}_m \neq x_m$ si apoi împărtiti la 999.

Problema 6.

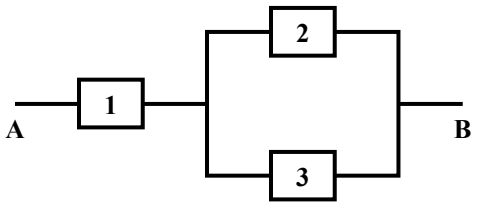
Fie Y o variabilă aleatoare caracterizată astfel: $\Pr(Y = 1) = 0{,}25$, $\Pr(Y = 2) = 0{,}25$, $\Pr(Y = 3) = 0{,}50$.

a. Reprezentati grafic asa-numita functie de probabilitate $p_Y(y)$.
b. Calculati $\Pr(Y < 1)$, $\Pr(Y \leq 1)$, $\Pr(Y > 2)$ si $\Pr(Y \geq 2)$.
c. Calculati $\Pr(1 \leq Y \leq 3)$, $\Pr(1 \leq Y < 3)$, $\Pr(1 < Y \leq 3)$ si $\Pr(1 < Y < 3)$.

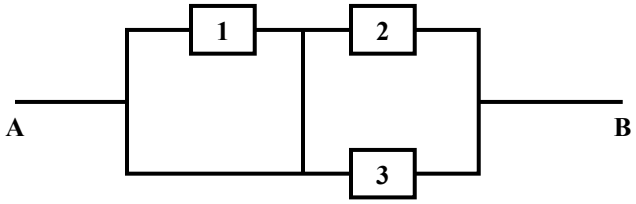
Problema 7.

Comutatoarele 1, 2, 3 operează corect cu probabilitătile 0,88, 0,92, respectiv 0,90. Ele operează independent.

a. Aflati probabilitatea ca un curent să circule de la A la B în circuitul de mai jos.



b. Aflati probabilitatea ca un curent să circule de la A la B în circuitul schitat mai jos.



Solutie:

a. Curentul circulă de la A la B dacă si numai dacă comutatoarele sunt în stările descrise în expresia "comutatorul 1 functional ŞI (comutatorul 2 functional SAU comuratorul 3 functional)" cu operatorul logic SAU înteles nu ca exclusivitate, nu ca un SAU EXCLUSIV. Starea perechii de comutatoare în paralel 2 si 3 contrară celei exprimate în paranteză este cea de nefunctionare concomitentă a lor. Probabilitatea acestei situatii, Pr(2 nefunctional) ŞI Pr(3 nefunctional) este $(1 - 0,92)\cdot(1 - 0,9) = 0,008$ dată fiind independenta functională a comutatoarelor. Starea cuprinsă în paranteza mentionată este probabilă în măsura $1 - 0,008 = 0,992$.
   Tot în virtutea independentei functionale a comutatoarelor, probabilitatea cerută este $0,88\cdot0,992 = 0,87296$.
b. Se observă că în circuit comutatorul 1 este de prisos si starea lui nu afectează în nici un fel "transparența" de la A la B. Rămân determinante stările comutatoarelor 2 si 3. La punctul anterior s-a stabilit că "transparența" este asigurată cu o probabilitate de 0,992.

Problema 8. Formula lui Bayes.

O companie producătoare de automobile produce marca Lăstun 2009 în patru locatii diferite: I, II, III si IV. Se presupune că 20% din acestea sunt produse la fabrica I, 23% sunt produse la fabrica II, 27% sunt produse la fabrica III si restul la fabrica IV. Se presupune de asemenea că 5% din masinile produse în locatia I trebuie rechemate pentru remedieri, la fel 6% din cele produse în locatia II, 3% din cele produse în locatia III si 8% din cele produse în locatia IV.

a. Care este probabilitatea ca un Lăstun 2009 ales la întâmplare să fie rechemat pentru remedieri?

b. Stiind că un automobil a fost rechemat pentru remedieri, calculati probabilitatea conditionată ca el să fie produs la fabrica I (II, III sau IV)
c. Stiind că un automobil nu trebuie rechemat pentru remedieri, calculati probabilitatea conditionată ca el să fie produs la fabrica I (II, III sau IV)

Problema 9.

Se dau două cutii. Cutia 1 contine 10 cărti din care 3 sunt marcate fiecare cu numărul "1" si celelalte 7 sunt marcate cu numărul "2". Cutia 2 contine 15 cărti dintre care 9 sunt marcate cu "1", iar restul de 6 sunt marcate cu "2".
Se execută următorul experiment aleator, în trei pasi. La pasul 1 se alege aleator echiprobabil o cutie si apoi se alege aleator o carte din acea cutie; numărul de pe carte (1 sau 2) este notat – să-i spunem numărul N_1. La pasul 2, se alege la întâmplare o carte din cutia N_1 si numărul de pe carte este si el notat – fie acesta N_2. La pasul 3 se alege tot aleator din cutia N_2 si numărul de pe carte se notează, de asemenea – fie acela N_3. Se consideră ca spatiu al esantioanelor pentru acest experiment multimea tuturor valorilor triple (N_1, N_2, N_3). Sunt 8 rezultate posibile în spatiul esantioanelor. Folositi teorema multiplicării ca ajutor în a calcula probabilitatea fiecăruia din cele 8 rezultate, în cazurile următoare:
a. În cazul când cartea este totdeauna pusă înapoi în cutia din care a fost aleasă înainte de alegerea unei alte cărti.
b. În cazul când o carte este totdeauna pusă înapoi, dar în cealaltă cutie.
c. Să presupunem că primul caz (a.) prevalează. Calculati probabilitătile Pr($N_3 = 1$) si Pr($N_3 = 2$).

Problema 10.

Se presupune că pachetele de biti sosesc la un server Internet de rutare la o rată medie de 2,3 pachete pe milisecundă.
a. Fie X numărul de pachete de mesaje care sosesc într-un interval de 10 milisecunde. Calculati probabilitătile Pr($X \geq 22$) si Pr($17 \leq X \leq 25$) (*Indicatie*: X este o variabilă aleatoare poissoniană).
b. Se presupune că serverul rutează pachetele pe care le primeste în modul următor: mai întâi intervalul este fragmentat în 10 subintervale de o milisecundă; apoi primul pachet sosit (dacă soseste vreunul) în fiecare subinterval de o milisecundă este rutat si pachetele sosite rămase nu sunt rutate (sunt distruse, de pildă). Fie Y numărul total de pachte rutate de server în intervalul de 10 milisecunde. Calculati Pr($Y = 8$) si Pr($6 < Y < 10$). (*Indicatie*: Y nu are o distributie Poisson ci are o distributie binomială).
c. Se presupune că serverul de rutare rutează pachetele pe care le primeste în 10 milisecunde în modul următor: mai întâi intervalul este fragmentat în 10 subintervale de o milisecundă; apoi primul si al doilea pachet care sosesc (dacă sosesc) în fiecare milisecundă sunt rutate, iar celelalte pachete care sosesc (dacă sosesc) sunt distruse. Fie Z numărul total de pachete rutate de server îtr-un interval de 10 milisecunde. Estimati Pr($Z = 16$) si Pr($14 \leq Z \leq$

18) utilizând 10.000 de observatii asupra lui Z simulate (Matlab). (Vor fi desigur niste valori estimate).

Problema 11.

Mary, Bill si Joe joacă jocul următor cu o monedă corectă (cel ce pierde dă un rând de beri): mai întâi Mary aruncă moneda – dacă obtine cap câstigă jocul; altminteri Bill aruncă si el moneda – dacă obtine cap câstigă jocul; altminteri Joe aruncă si el moneda – dacă obtine cap câstigă jocul. Dacă nimeni nu câstigă jocul la primul tur de aruncări se procedează la alte tururi identice cu primul, până când cineva câstigă jocul.

a. Fie Z o variabilă aleatoare geometrică cu parametrul $p = 1/2$. Stabiliti o partitie de întregi pozitivi în trei submultimi E_1, E_2, E_3 astfel încât probabilitătile evenimentelor $\{Z \in E_1\}$, $\{Z \in E_2\}$, $\{Z \in E_3\}$ să măsoare sansele ca jocul să fie câstigat de Mary, de Bill, respectiv de Joe.
b. Calculati probabilitătile ca Mary să câstige jocul, ca Bill să câstige jocul, ca Joe să câstige jocul, respectiv, uzând de distributia geometrică.

Problema 12. Functii de repartitie

Se generează un tabel de valori ale unei functii de repartitie si se utilizează pentru a calcula eficient probabilităti.

a. Rulati programul Matlab care urmează pentru a genera un vector de valori ale functiei de repartitie pentru o variabilă aleatoare X distribuită binomial cu parametrii $n = 13$ si $p = 0{,}3$.

```
s=1;
n=13;
p=0.3;
for i=1:n
      v1=[0 s];
      v2=[s 0];
      s=p*v1+(1-p)*v2;
end
cdf_values=cumsum(s);
```

Afisati sub formă de tabel cele 14 valori ale functiei de repartitie $F_X(x)$ pentru $x = 0, 1, 2, \ldots, 13$. Calculati probabilitatea $\Pr(X > 2)$ printr-o singură căutare în tabel. Calculati probabilitatea $\Pr(2 \leq X \leq 5)$ prin numai două căutări în tabel.

b. Executati programul Matlab următor pentru a genera un vector cu primele 18 valori ale functiei de repartitie a variabilei aleatoare Y distribuită după o lege Poisson cu parametrul $\lambda = 5{,}7$.

```
lambda=5.7;
s(1)=1;
for k=1:17
```

```
      s(k+1)=lambda*s(k)/k;
end
y=exp(-lambda)*s;
cdf_values=cumsum(y);
```

Afisati sub formă de tabel cele 18 valori ale functiei $F_Y(y)$ pentru $y = 0, 1, 2, \ldots, 17$. Calculati probabilitatea $\Pr(X \geq 4)$ printr-o singură căutare în tabel. Calculati probabilitatea $\Pr(2 < X < 5)$ prin numai două căutări în tabel.

Problema 13.

Fie U o variabilă aleatoare distribuită uniform pe intervalul [2, 13]. Utilizând sintaxa Matlab, se poate defini o variabilă aleatoare Z astfel:

$$Z = (U < 4) + (U < 7) + (U < 8) + (U < 11)$$

Z este o variabilă aleatoare discretă care ia valorile 0, 1, 2, 3, 4. Stabiliti functia de probabilitate pentru variabila Z (calculati $p_Z(z)$ pentru $z = 0, 1, 2, 3, 4$).

Solutie: Din enunt se întelege că evenimentele $(U < a)$ generează un 1 sau un 0 după cum ele se produc sau nu. Astfel, pentru $(U \geq 11)$, $Z = 0$ deoarece nici unul dintre cele patru evenimente nu se produce. Se mai observă faptul că unele dintre evenimente le pot implica pe altele. De pildă $(U < 4)$ le implică pe toate celelalte. Si acum calculul probabilităților cerute:

$$p_Z(0) = \Pr(U \geq 11) = 2/11$$
$$p_Z(1) = \Pr[(8 \leq U < 11) = 3/11$$
$$p_Z(2) = \Pr[(7 \leq U < 8) = 1/11$$
$$p_Z(3) = \Pr[(4 \leq U < 7) = 3/11$$
$$p_Z(4) = \Pr[U < 4) = 2/11$$

Problema 14. Distributia exponentială si distributia gaussiană

a. Fie X o variabilă aleatoare distribuită exponential cu parametrul $\lambda = 0{,}5$. Calculati $\Pr(1 \leq X \leq 2)$ si $\Pr(X \geq 1{,}5)$ prin integrarea functiei densitate de probabilitate.
b. Se dă o variabilă aleatoare normală (gaussiană) T cu media 10 si dispersia 225. Calculati $\Pr(T > 32)$, $\Pr(T < 0)$ si $\Pr(T > 60)$ în două moduri: (i) prin utilizarea tabelelor cu valori ale functiei de repartitie gaussiene; (ii) prin utilizarea functiei Matlab **`erf`**.

Problema 15. Cuantizor

Fie X o variabilă aleatoare continuă cu densitatea de probabilitate $f_X(x)$ de forma

$$f_X(x) = \begin{cases} C(3 - |x|) & -3 \leq x \leq 3 \\ 0 & \text{în rest} \end{cases}$$

cu C o constantă pozitivă. Forma generală a unui cuantizor pentru X, $Q(x)$, pe două niveluri este

$$Q(x) = \begin{cases} -B & -3 \le x \le 0 \\ B & 0 < x \le 3 \end{cases}$$

cu B o constantă între 0 si 3.

a. Determinati valoarea unică C care face din $f_X(x)$ o densitate de probabilitate.
b. Se pune în cuantizorul $Q(x)$, $B = 1{,}5$ (astfel B este punctul median al intervalului [0, 3] si $Q(x)$ se numeste cuantizorul uniform pe două niveluri pentru X). Calculati $E[(X - Q(x))^2 | -3 \le X \le 0]$, $E[(X - Q(x))^2 | 0 < X \le 3]$ si $E[(X - Q(x))^2]$. (Cantitatea ultimă este *eroarea de cuantizare medie pătratică* rezultată la utilizarea lui $Q(x)$ pentru a cuantiza pe X.).
c. Calculati $E[X | 0 < X \le 3]$. Acum luati $B = E[X | 0 < X \le 3]$ în cuantizorul $Q(x)$. Calculati $E[(X - Q(x))^2 | -3 \le X \le 0]$, $E[(X - Q(x))^2 | 0 < X \le 3]$ si $E[(X - Q(x))^2]$. Este eroarea de cuantizare $E[(X - Q(x))^2]$ mai mică decât cea calculată la punctul b.? (Ar trebui să fie).

Solutie:

a. O functie densitate de probabilitate trebuie să verifice egalitatea:

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

Din această ecuatie rezultă valoarea lui $C = 1/9$.

b. Mediile cerute sunt următoarele:

$$E[(X - Q(x))^2 | -3 \le X \le 0] = 2\int_{-3}^{0}\left(x + \frac{3}{2}\right)^2 \frac{1}{9}(3 + x)dx = \frac{3}{4}$$

$$E[(X - Q(x))^2 | 0 < X \le 3] = 2\int_{0}^{3}\left(x - \frac{3}{2}\right)^2 \frac{1}{9}(3 - x)dx = \frac{3}{4}$$

$$E[(X - Q(x))^2] = \int_{-3}^{0}\left(x + \frac{3}{2}\right)^2 \frac{1}{9}(3 + x)dx + \int_{0}^{3}\left(x - \frac{3}{2}\right)^2 \frac{1}{9}(3 - x)dx = \frac{3}{4}$$

Coeficientul 2 pentru primele două medii provine de la conditionarea repartitiilor: valorile negative si pozitive sunt egal probabile, probabilitătile asociate sunt ambele egale cu 1/2.

c. Cu media cerută

$$B = 2\int_{0}^{3}\frac{1}{9}x(3 - x)dx = 1$$

calculele de la punctul anterior se repetă cu această nouă valoare:

$$E[(X - Q(x))^2 | -3 \le X \le 0] = 2\int_{-3}^{0}(x + 1)^2 \frac{1}{9}(3 + x)dx = \frac{1}{2}$$

$$E[(X - Q(x))^2 | 0 < X \le 3] = 2\int_{0}^{3}(x - 1)^2 \frac{1}{9}(3 - x)dx = \frac{1}{2}$$

$$E[(X - Q(x))^2] = \int_{-3}^{0}(x + 1)^2 \frac{1}{9}(3 + x)dx + \int_{0}^{3}(x - 1)^2 \frac{1}{9}(3 - x)dx = \frac{1}{2}$$

Într-adevăr, eroarea de cuantificare este de data aceasta mai mică.

Problema 16.

Fie X o variabilă aleatoare binomială cu $n = 45$ si $p = 1/3$ si fie Y o variabilă aleatoare poissoniană cu $\lambda = 10$. Se presupune că cele două variabile sunt corelate într-un mod necunoscut (adică X si Y nu sunt independente). Se presupune că din suficient de multe observatii ale valorilor X si Y s-a stabilit cu un înalt grad de încredere că $\text{Var}(X + Y) = 200$.

a. Care sunt mediile si dispersiile pentru fiecare din variabilele aleatoare X si Y?
b. Calculati $E(XY)$. (*Indicatie*: Se scrie $\text{Var}(X + Y) = E[(X + Y)^2] - [E(X) + E(Y)]^2$ etc.)

Problema 17.

O variabilă aleatoare R are functia densitate de repartitie următoare (distributia Rayleigh)

$$f_R(r) = \begin{cases} re^{-r^2/2} & r \geq 0 \\ 0 & \hat{i}n\ \ rest \end{cases}$$

Media si dispersia sunt date de relatiile $E[R] = \sqrt{\pi/2}$, $\text{Var}[R] = 2 - \pi/2$.

a. Dacă U este o variabilă aleatoare uniform distribuită între 0 si 1, stabiliti o transformare $R = \phi(U)$ astfel încât R să aibă densitatea de probabilitate $f_R(r)$.
b. Utilizând transformarea de la punctul a., scrieti un program Matlab simplu care să genereze un vector de 10.000 observatii simulate ale valorilor lui R. Utilizati functiile Matlab **`mean`** si **`var`** pe vectorul generat pentru a obtine estimatii ale mediei $E[R]$ si dispersiei $\text{Var}[R]$ date mai sus. Sunt estimările obtinute estimări bune?
c. Se cunoaste că dacă variabilele aleatoare X, Y sunt repartizate normal cu media 0 si dispersia 1, atunci $R = \sqrt{X^2 + Y^2}$ are densitatea de probabilitate $f_R(r)$. Uzând de acest fapt, scrieti un alt program simplu Matlab care să genereze un vector de 10.000 de observatii simulate asupra lui R. Utilizati functiile Matlab **`mean`** si **`var`** pe vectorul generat pentru a obtine estimatii ale mediei $E[R]$ si dispersiei $\text{Var}[R]$ date mai devreme. Probabil aceste estimări vor fi cam aceleasi cu cele de la punctul b.

Problema 18. Varabile aleatoare mixte

Fie D o variabilă aleatoare mixtă, cu $D = 0$ o valoare cu probabilitate proprie α ($0 < \alpha < 1$) si cu o densitate de probabilitate exponentială pentru $D > 0$, nulă pentru $D < 0$.
Calculati $E(D)$ si $\text{Var}(D)$.
Indicatie: Calea cea mai facilă de a rezolva această problemă trece prin formulele

$$E[D] = E[D|D = 0]\Pr(D = 0) + E[D|D > 0]\Pr(D > 0)$$

$$E[D^2] = E[D^2|\mathrm{D} = 0]\Pr(D = 0) + E[D^2|D > 0]\Pr(D > 0)$$

SURSE DE INFORMATIE

Lucrarea 1

Tema 1: Entropia surselor de informatie fără memorie.

Sursele studiate au un alfabet care poate cuprinde de la 2 la 15 simboluri si numărul *n* al simbolurilor este limitat prin program.
Alegerea numărului de simboluri se face intervenind în program: linia în care se poate face modificarea este marcată cu un comentariu potrivit. Se poate interveni si asupra limitelor 2, 15, dar această interventie nu aduce în discutie aspecte noi si de aceea nu este recomandată în mod special.
Simbolurile sursei sunt considerate a fi chiar numerele de la 1 la *n*.
În prima fază este evaluată *entropia maximă* a unei surse cu un alfabet alcătuit din *n* simboluri, entropie care se atinge atunci când simbolurile sunt echiprobabile.
În faza a doua se generează o listă de probabilitati asociate celor *n* simboluri arbitrară. Cum este de asteptat, *entropia* sursei este totdeauna inferioară entropiei maxime.
Diferenta dintre entropia maximă si cea efectivă este *redundanta* sursei.
Programul evaluează entropia efectivă si redundanta sursei.
Asupra sursei se fac succesiv unele modificări (trei).
Prima modificare: Se aleg la întâmplare două simboluri ale sursei. Uzual acestea au probabilităti de aparitie diferite. Se face o medie aritmetică a celor două probabilităti si se atribuie ca valori noi probabilităților celor două simboluri selectate. Prin aceasta se aduce o uniformizare (partială) în lista de probabilităti. Efectul: cresterea entropiei.
A doua modificare: Se reduce numărul de simboluri de la *n* la *n* – 1. Penultimul simbol din sursa originară se comasează cu ultimul si probabilitătile lor se adună. Penultimul simbol devine ultimul. Efectul: scăderea entropiei.
A treia modificare: Se adaugă la lista de *n* – 1 simboluri ale sursei precedente un nou simbol, al *n*-lea, cu probabilitatea de aparitie nulă. Efectul: entropia rămâne aceeasi.
Programul apelează ocazional functia **`entropie`**, dată si ea în listingul alăturat. Functia aceata nu este altceva decât transcrierea în Matlab a formulei binecunoscute pentru entropia unei surse simple fără memorie.
Se recomandă executarea repetată a programului, cu acelasi număr *n* de simboluri, cu valori *n* diferite. A se observa entropiile, redundantele, efectele unor particularităti ale sursei.

```
clear
n=8;    % numarul de simboluri ale sursei
if n>15
```

```
    n=15;
end    % o limitare a numarului de simboluri
if n<2
    n=2;
end    % o alta limitare a numarului de simboluri
simb=1:n;    % simbolurile sursei
prob=(1/n)*ones(1,n);
entropia_maxima=entropie(prob)
prob=rand(1,n);    % generarea a n numere aleatoare intre 0 si 1
prob=prob/sum(prob);    % normaliarea listei de n numere; suma =
1
simboluri=simb
probabilitati=prob
h=entropie(prob);
entropia_efectiva=h
redundanta_sursei=entropia_maxima-entropia_efectiva
display('*** Uremaza o egalizare a probabilitatilor a doua
simboluri ***')
display('*** Pentru continuare apasati Enter! ***')
pause
r=ceil(n*rand(1,2))    % se genereaza doua numere de la 1 la n
med=(prob(r(1))+prob(r(2)))/2;    % se mediaza probabilitatile
prob(r(1))=med;
prob(r(2))=med;
simboluri=simb
probabilitati=prob
h=entropie(prob);
entropia_noua=h
display('*** Uremaza o reducere a numarului de simboluri ***')
display('*** Pentru continuare apasati Enter! ***')
pause
prob(n-1)=prob(n-1)+prob(n);    % se pun laolalta ultimele doua
simboluri
simboluri=simboluri(1:(n-1))
probabilitati=prob(1:(n-1))
h=entropie(prob(1:(n-1)));
entropia_noua=h
display('*** Uremaza adaugarea unui simbol cu probabilitate nula
***')
display('*** Pentru continuare apasati Enter! ***')
pause
prob(n)=0;    % se adauga un simbol cu probabilitatea nula
simboluri=[simboluri n]
probabilitati=prob
h=entropie(prob);
entropia_noua=h

function entropie=ent(s)
% s este vectorul (linie) al probabilitatilor
[n,m]=size(s);
ent=0;
for i=1:m
    if s(i)>0
        ent=ent-s(i)*log(s(i));
```

```
    end
end
ent=ent/log(2);
entropie=ent;
```

Tema 2: Entropia surselor duble de informatie

Programul Matlab care urmează se ocupă de caracterizarea surselor de informatie pereche de surse simple (A, B), notate si $A \times B$, independente sau dependente una de cealaltă. Sursele acestea privite unitar generează simboluri proprii, perechi ordonate de simboluri (a, b), unul din alfabetul sursei A, altul din alfabetul sursei B.
Dacă sursele simple componente sunt independente, atunci probabilitătile simbolurilor (a, b) se obtin prin multiplicarea probabilităților asociate simbolurilor din pereche. Dacă sursele A, B sunt dependente atunci intră în rol formula probabilităților conditionate. Pentru calculul probabilității unei perechi (a, b), se multiplică probabilitatea lui a cu probabilitatea lui b conditionată de a sau se multiplică probabilitatea lui b cu probabilitatea lui a conditionată de b.
Toate aceste relatii se pot regăsi în programul alăturat.
Programul începe cu alocarea dimensiunilor m si n ale celor două multimi de simboluri (multimi alfabetice) pentru cele două surse A si B.
În prima parte a programului, sursele A si B sunt considerate independente. Rezultatul calculelor este: entropia sursei $A \times B$ este egală cu suma entropiilor surselor A si B.
În partea a doua, sursele A si B se fac dependente. Sursa A se mentine în forma initializată în partea introductivă a programului. Sursa B nu mai poate fi arbitrară, cea deja utilizată. Sursa B, chiar dacă numărul de simboluri rămâne neschimbat are o listă de probabilităti care rezultă din conditionarea (dependența) mutuală a celor două surse simple. Rezultatul ultim al calculelor este: entropia sursei $A \times B$ este mai mică decât suma entropiilor surselor A si B luate separat.

```
clear
m=7;
n=5;    % se alege numarul simbourilor generate de cele doua
surse
proa=rand(1,m);    % generarea a m numere aleatoare intre 0 si 1
proa=proa/sum(proa);    % normaliarea listei de m numere; suma =
1
prob=rand(1,n);    % generarea a n numere aleatoare intre 0 si 1
prob=prob/sum(prob);    % normaliarea listei de n numere; suma =
1
disp(' ')
s='Surse independente, probabilitati, entropii';
disp(s)
A=proa
B=prob
a=entropie(proa);    % entropia sursei A
```

```
b=entropie(prob);    % entropia sursei B
Entropii=[a b]
Suma_entropiilor=a+b    % suma entropiilor
prod=proa'*prob;    % probabilitatile perechilor (a,b), A, B
independente
AXB=prod
for i=1:m
    entrl(i)=entropie(prod(i,:));
end
entropie_AXB=sum(entrl)    % entropia sursei duble AXB, A, B
independnete
disp(' ')
s='Surse dependente, probabilitati, entropii';
disp(s)
cond=rand(m,n);
for i=1:m
    cond(i,:)=cond(i,:)/sum(cond(i,:));
end    % generarea probabilitatilor conditionate P(b/a)
disp(' ')
s='Matricea de conditionare b/a';
disp(s)
cond
for i=1:n
    prod(:,i)=proa'.*cond(:,i);
end    % calculul probabilitatilor pe perechi (a,b), A, B
dependente
prob=sum(prod);    % calculul probabilitatilor pentru sursa B
B=prob
b=entropie(prob)    % entropia sursei B
Entropii=[a b]
Suma_entropiilor=a+b    % suma entropiilor
AXB=prod
for i=1:m
    entrl(i)=entropie(prod(i,:));
end
entropie_AXB=sum(entrl)    % entropia sursei duble AXB, A, B
independente
```

Lucrarea 2

Tema: Entropia surselor de informatie markoviene.

Sursele Markov propuse spre studiu au de la 2 la 8 stări si numărul *n* al stărilor este limitat prin program.
Alegerea numărului de stări se face intervenind în program: linia în care se poate face modificarea este marcată cu un comentariu potrivit. Se poate interveni si asupra limitelor 2, 8, dar această interventie nu aduce aspecte noi si de aceea nu se recomandă în mod special.
Se generează aleator o matrice a probabilităților de tranzitie din cele *n* stări în cele *n* stări ale sursei. Matricea *p* nu poate fi decât pătrată, $n \times n$, cu suma liniilor egală cu unitatea.

Starea initială *s* a sursei este precizată uzual printr-un vector (linie) de probabilităti, cu suma componentelor egală cu 1. În program, starea initială poate fi modificată în linia/liniile unde ea este definită, marcată si ea cu un comentariu. Starea initială, ca si oricare altă stare ulterioară este caracterizată de o entropie care este evaluată cu functia **`entropie`** reprodusă într-un listing aparte, la finalul acestor câteva pagini de îndrumare. Functia **`entropie`** nu este altceva decât transcrierea în Matlab a formulei binecunoscute pentru entropia unei surse simple (fără memorie).
Înainte de prima pauză în executia programului, se afisează matricea de tranzitie, starea initială si entropia corespunzătoare.
Apoi se simulează o evolutie a sursei (10 pasi). De regulă, sursa evoluează către o stare stationară. Se afisează stările succesive si entropiile asociate. Se poate observa o convergentă către o stare numită si *stare stationară*. Această stare nu este explicită, este numai bănuită din evolutia sursei. Lucrurile devin ceva mai clare dacă se extinde evolutia la mai mult de 10 pasi, prin modificarea limitei superioare a instructiunii "`for k=1:10`".
După următoarea pauză în executie se calculează efectiv starea stationară. Rezultatele afisate diferă întrucâtva de ultima stare din secventa evolutivă anterioară. De regulă, starea stationară este descrisă printr-un vector de probabilităti cu toate componentele nenule. Se zice în aceste cazuri că sursa este *ergodică*. Uneori, un vector al unei stări stationare contine componente nule (stări excluse pe termen lung) sau o componentă unitară si celelalte nule (stări absorbante). Aceste situatii caracterizează surse *neergodice*.
Se recomandă executarea repetată a programului, cu acelasi număr *n* de stări sau cu valori ale lui *n* diferite. A se observa evolutia sursei, stările si entropiile asociate, starea stationară.

```
clear
n=4;    % numarul de stari ale sursei
if n>8
    n=8;
end    % o limitare a numarului de stari
if n<2
    n=2;
end    % o alta limitare a numarului de stari
p=rand(n);
for i=1:n
    suml=sum(p(i,:));
    p(i,:)=p(i,:)/suml;
end    % crearea unei matrici a tranzitiilor
matricea_tranzitiilor=p
s=zeros(1,n);
s(2)=1;    % starea initiala
starea_initiala_si_entropia=[s entropie(s)]
display('*** Uremaza o evolutie a sursei Markov (10 pasi) ***')
display('*** Pentru continuare apasati Enter! ***')
pause
for k=1:10
```

```
    s=s*p;
    starea_curenta_si_entropia=[s entropie(s)]
end
display('*** Uremaza calculul starii stationare a sursei Markov
***')
display('*** Pentru continuare apasati Enter! ***')
pause
a=p-eye(n);
a=a';
ap=zeros(n-1);
for i=1:n
    k=0;
    for j=1:n
        if j~=i
            k=k+1;
            ap(1:(n-1),k)=a(2:n,j);
        end
    end
    compl(i)=(-1)^i*det(ap);
end    % evaluarea starii stationare
s=compl/sum(compl);    % probabilitatile starii stationare
starea_stationara=s
starea_stationara=s*p    % verificarea starii stationare
starea_stationara_si_entropia=[starea_stationara entropie(s)]

function entropie=ent(s)
% s este vectorul (linie) al probabilitatilor
[n,m]=size(s);
ent=0;
for i=1:m
    if s(i)>0
        ent=ent-s(i)*log(s(i));
    end
end
ent=ent/log(2);
entropie=ent;
```

Lucrarea 3

Tema: Entropia surselor de informatie Markov binare.

Sursele Markov binare studiate au 2, 4, 8 sau 16 stări, numărul stărilor *n* este totdeauna o putere a lui 2 si este limitat prin program. Exponentul este ordinul sursei.

Alegerea ordinului se face intervenind în program: linia în care se poate face modificarea este marcată de un comentariu adecvat. Se poate interveni si asupra limitelor 2 si 16, dar această interventie nu aduce aspecte noi si de aceea nu este recomandată special.

Se generează aleator o matrice a probabilităților de tranzitie din cele *n* stări în cele *n* stări ale sursei. Matricea *p* nu poate fi decât pătrată, $n \times n$, cu suma liniilor egală cu unitatea. Multe elemente în această matrice pot fi si chiar sunt nule. De

fapt, pe fiecare linie numai două elemente sunt nenule deoarece numai două tranzitii sunt posibile, din starea de plecare în numai alte două stări care se obtin prin deplasarea spre stânga a expresiei binare a stării curente si înlocuirea ultimului bit cu 0 sau 1. De pildă, din starea curentă 101 nu se poate ajumge decât în stările 010 sau 011 (s-a subliniat de fiecare dată partea rămasă din expresia binară a stării curente după deplasarea la stânga cu pierderea primului bit). O trecere din starea 101 în alte stări, de pildă în starea 110, nu este posibilă într-un singur pas.
Starea initială *s* a sursei este precizată printr-un vector (linie) de probabilităti, cu suma componentelor egală cu 1. În program, starea initială poate fi modificată în linia/liniile unde ea este definită, marcată dealtfel cu un comentariu. Starea initială, ca si oricare altă stare ulterioară este caracterizată de o entropie care este evaluată cu functia **`entropie`**, dată si ea în listingul alăturat. Functia aceasta nu este altceva decât transcrierea în Matlab a formulei binecunoscute pentru entropia unei surse simple (fără memorie).
Înainte de prima pauză în executia programului, se afisează matricea de tranzitie, starea initială si entropia corespunzătoare.
Apoi se simulează o evolutie a sursei (10 pasi). De regulă, sursa evoluează către o stare stationară. Se afisează stările succesive si entropiile asociate. Se poate observa o convergentă către o stare numită si *stare stationară*. Această stare nu este explicită, este numai bănuită din evolutia sursei. Lucrurile devin ceva mai clare dacă se extinde evolutia la mai mult de 10 pasi, prin modificarea limitei superioare pentru `k` în instructiunea "`for k=1:10`".
După următoarea pauză în executie se calculează efectiv starea stationară. Rezultatele afisate diferă întrucâtva de ultima stare din secventa evolutivă anterioară. De regulă, starea stationară este descrisă printr-un vector de probabilităti cu toate componentele nenule. Se zice în aceste cazuri că sursa este *ergodică*. Uneori, un vector al unei stări stationare contine componente nule (stări excluse pe termen lung) sau o componentă unitară si celelalte nule (stări absorbante). Aceste situatii caracterizează sursele *neergodice*.
Se recomandă executarea repetată a programului, pentru surse de acelasi ordin sau de ordine diferite. A se observa evolutia sursei, stările si entropiile asociate, starea stationară.

```
clear
ordin=2;    % ordinul sursei
n=2^ordin;    % numarul de stari ale sursei
if n>16
    n=16;
end    % o limitare a numarului de stari
if n<2
    n=2;
end    % o alta limitare a numarului de stari
for i=1:n
    s=dec2bin(i-1,ordin);
    stari(i,:)=s;
    starin(i)=i-1;
```

```
end
stari
p=zeros(n);
for i=1:n
    m=mod(2*(i-1),n);
    m1=m;
    m2=mod(m+1,n);
    p(i,m1+1)=rand;
    p(i,m2+1)=rand;
end
for i=1:n
    suml=sum(p(i,:));
    p(i,:)=p(i,:)/suml;
end    % crearea unei matrici a tranzitiilor
matricea_tranzitiilor=p
s=zeros(1,n);
s(1)=1;    % starea initiala
starea_initiala_si_entropia=[s entropie(s)]
display('*** Uremaza o evolutie a sursei Markov (10 pasi) ***')
display('*** Pentru continuare apasati Enter! ***')
pause
for k=1:10
    s=s*p;
    starea_curenta_si_entropia=[s entropie(s)]
end    % evolutia sursei
display('*** Uremaza calculul starii stationare a sursei Markov
***')
display('*** Pentru continuare apasati Enter! ***')
pause
a=p-eye(n);
a=a';
ap=zeros(n-1);
for i=1:n
    k=0;
    for j=1:n
        if j~=i
            k=k+1;
            ap(1:(n-1),k)=a(2:n,j);
        end
    end
    compl(i)=(-1)^i*det(ap);
end    % evaluarea starii statinare
s=compl/sum(compl);    % probabilitatile starii stationare
starea_stationara=s
starea_stationara=s*p    % verificarea starii stationare
starea_stationara_si_entropia=[starea_stationara entropie(s)]

function entropie=ent(s)
% s este vectorul (linie) al probabilitatilor
[n,m]=size(s);
ent=0;
for i=1:m
    if s(i)>0
        ent=ent-s(i)*log(s(i));
```

```
    end
end
ent=ent/log(2);
entropie=ent;
```

Problema 19.

O monedă perfectă este aruncată succesiv până apare prima stemă. Fie X numărul de aruncări necesare.

a. Aflati entropia $H(X)$ în biti. Expresiile următoare pot fi utile în evaluarea cerută:

$$\sum_{n=1}^{\infty} r^n = \frac{r}{1-r}, \sum_{n=1}^{\infty} nr^n = \frac{r}{(1-r)^2}$$

b. Potrivit acestei scheme se generează o variabilă aleatoare X. Stabiliti o secventă eficientă de întrebări cu răspunsuri da-nu, de forma "Este X continut în multimea **S**?". Comparati $H(X)$ cu numărul mediu de întrebări necesare pentru a determina X.

Solutie.

a. La prima aruncare, probabilitatea de a încheia experimentul este 1/2. Probabilitatea de a încheia experimentul la a doua aruncare este 1/4. Probabilitatea de a obtine prima oară stemă la a n-a aruncare este $(1/2)^n$. Variabila aleatoare X este de tip discret cu probabilitătile asociate numerelor naturale egale cu puterile succesive ale lui 1/2. Entropia cerută este dată de formula

$$H(X) = \sum_{n=1}^{\infty} \frac{1}{2^n} \log 2^n = \sum_{n=1}^{\infty} n\left(\frac{1}{2}\right)^n = \frac{1/2}{(1-1/2)^2} = 2$$

b. Ideea este a obtine maximum de informatie la fiecare întrebare. Fie a un număr care partitionează prin ordine ($n \le a$, $n > a$) multimea numerelor naturale în multimea **S** si multimea **N** – **S**. Întrebarea "Este X continut în multimea **S**?" poate avea răspunsul "da" cu probabilitatea $\sum_{n=1}^{k} \frac{1}{2^n} = 1 - \frac{1}{2^k}$, "nu" cu probabilitatea $\sum_{n=k+1}^{\infty} \frac{1}{2^n} = \frac{1}{2^k}$ $(a - 1 < k \le a)$.

Informatia medie obtinută la o întrebare de acest gen este

$$-\left(1-\frac{1}{2^k}\right)\log\left(1-\frac{1}{2^k}\right) - \frac{1}{2^k}\log\frac{1}{2^k}$$

etc.

Problema 20. Regula grupării pentru entropie.

Fie $p = (p_1, p_2, \ldots, p_m)$ o distributie de probabilităti pentru m elemente, adică un vector de m numere nenegative care însumate dau unitatea. Se defineste o distributie nouă, q pentru $m - 1$ elemente conform schemei $q_1 = p_1$, $q_2 = p_2$, …, $q_{m-2} = p_{m-2}$ si $q_{m-1} = p_{m-1} + p_m$, cu alte cuvinte distributia nouă este aceeasi cu

cea initială pentru indici de la 1 la $m - 2$ si probabilitatea q_{m-1} se obtine prin însumarea ultimelor două valori din lista de probabilităti initială. Arătati că

$$H(p) = H(q) + (p_{m-1} + p_m) H_2\left(\frac{p_{m-1}}{p_{m-1} + p_m}, \frac{p_m}{p_{m-1} + p_m}\right)$$

unde $H_2(a, b) = -a \log a - b \log b$.

Problema 21.

Fie X o variabilă aleatoare discretă. Arătati că entropia unei functii de X este inferioară cel mult egală cu entropia lui X prin justificarea pasilor următori:

$$H(X, g(X)) = H(X) + H(g(X)/X) = H(X)$$
$$H(X, g(X)) = H(g(X)) + H(X/g(X)) \geq H(g(X))$$

prin urmare $H(g(X)) \leq H(X)$.

Solutie:

$H(X, g(X)) = H(X) + H(g(X)|X)$ rezultă din simpla utilizare a unei formule pentru entropii discutate la curs.

$H(g(X)|X) = 0$ deoarece pentru orice valoare a lui X, $g(X)$ este univoc deteminată asa încât $H(g(X)|X) = \Sigma_x p(x) H(g(X)|X = x) = \Sigma_x 0 = 0$.

$H(X, g(X)) = H(g(X)) + H(X|g(X))$ rezultă tot asa prin simpla utilizare a aceleiasi formule pentru entropii discutată la curs.

$H(X|g(X)) \geq 0$, cu egalitate dacă si numai dacă X este o functie de g(X), cu alte cuvinte functia $g(\cdot)$ este o corespondentă biunivocă. Asadar, $H(X, g(X)) \geq H(g(X))$.

Prin combinarea afirmatiilor a doua si a patra, se obtine $H(g(X)) \leq H(X)$.

Problema 22. Functii

a. Fie $Y = X^5$, cu X o variabilă aleatoare care ia valori pozitive si valori negative. Care este relatia între entropiile $H(X)$ si $H(Y)$?
b. Dar dacă $Y = X^2$?
c. Dar dacă $Y = \text{tg}X$?

Solutie: Din problema precedentă, se stie că prin trecerea unei variabile aleatoare printr-o functie nu se poate decât reduce entropia sau poate fi, în cazul biunivocitătii, mentinută la aceeasi valoare. Ea niciodată nu creste. Asadar, $H(g(X)) \leq H(X)$, pentru orice functie g. Ratiunea este simplă: dacă functia g nu este biunivocă, atunci ea pune laolaltă unele stări si prin aceasta reduce entropia.

Solutia acestei probleme rezidă în a determina de la caz la caz dacă este vorba sau nu de o aplicatie bijectivă. De observat că bijectivitatea se referă exclusiv la suportul variabilei X, adică la acele valori x pentru care $p(x) > 0$.

a. $Y = X^5$ este o bijectie, asadar entropia este o functie numai de parobabilităti (si nu de valorile rezultatelor observate) si de aceea nu se schimbă: $H(X) = H(Y)$.

b. $Y = X^2$ este o functie pară si, în consecintă, nu este o corespondentă biunivocă cu exceptia cazului în care suportul variabilei X nu contine simultan valori x si opusele lor $-x$.
c. Ca si la punctul a., $Y = \mathrm{tg}X$ este o bijectie (ca functie pe întregi) si $H(X) = H(Y)$.

În general, pentru punctul b., $H(Y) \le H(X)$. Pentru acest caz este posibil a se da o margine superioară pentru $H(X)$: deoarece X poate lua numai două valori la orice Y dat, $H(X|Y) \le \log 2 = 1$. Asadar

$$H(Y) \le H(X) = H(X, Y) = H(Y) + H(X|Y) \le H(Y) + 1$$

Problema 23. Bytes (octeti)

Entropia, $H_a(X) = -\Sigma p(x)\log_a p(x)$ se exprimă în biti dacă logaritmul este în baza 2 si în bytes (octeți) dacă logaritmul este în baza 256. Care este relația între $H_2(X)$ si $H_{256}(X)$?

Solutie:

$$H_2(X) = -\sum p(x)\log_2 p(x) = -\sum p(x)\frac{\log_2 p(x)\log_{256} 2}{\log_{256} 2} =$$

$$= -\sum p(x)\frac{\log_{256} p(x)}{\log_{256} 2} = -\frac{1}{\log_{256} 2}\sum p(x)\log_{256} p(x) = \frac{H_{256}(X)}{\log_{256} 2}$$

Astfel

$$H_2(X) = 8H_{256}(X)$$

Problema 24. Cântărirea monedelor

Se admite existența a n monede între care una ar putea fi contrafăcută, falsă. Dacă există o monedă falsă, ea poate fi sau mai grea, sau mai usoară decât celelalte. Monedele urmează a fi cântărite cu o balanță.

a. Numărati stările în care pot fi cele n monede si numărati rezultatele cântăririlor în număr de k. Prin comparare, evaluați o limită superioară a numărului de monede n astfel încât prin k operații de cântărire să se găsească moneda falsă (dacă ea există) si să se precizeze dacă ea este mai grea sau este mai usoară.
b. (*Mai dificil, deci optional*) Care este strategia potrivită pentru $k = 3$ cântăriri si $n = 12$ monede?

Solutie:

a. Pentru n monede, sunt $2n + 1$ situatii posibile sau stări.
   - Una din cele n monede este mai grea;
   - Una din cele n monede este mai usoară;
   - Toate au aceeasi greutate.

   Fiecare cântărire poate avea unul din trei rezultate posibile: egalitate, talerul din stânga mai greu sau talerul din dreapta mai greu. În k operații de cântărire, sunt posibile 3^k rezultate ceea ce face posibilă distincția între cel mult 3^k "stări" diferite. Asadar, $2n + 1 \le 3^k$, sau $n \le (3^k - 1)/2$.

Din punct de vedere al informației, fiecare cântărire aduce cel mult $\log_2 3$ biti de informatie. Sunt $2n + 1$ "stări" posibile, cu un maxim de entropie de $\log_2(2n + 1)$ biti. În consecință, în această situație sunt necesare cel puțin $\log_2(2n + 1)/\log_2 3$ cântăriri pentru a extrage suficientă informatie despre moneda falsă. Se observă, rezultatul este acelasi ca acela de mai devreme.

b. Această problemă cunoaste mai multe abordări posibile. Una din ele se bazează pe sistemul de numeratie ternar.
Numerele {–12, –11, …, –1, 0, 1, …, 12} se pot exprima într-un sistem de numeratie ternar cu alfabetul {–1, 0, 1}. De pildă numărul 8 este (–1)01 deoarece $(-1)\times 3^0 + 0\times 3^1 + 1\times 3^2 = 8$. Se formează o matrice cu coloanele reprezentând numere pozitive

	1	2	3	4	5	6	7	8	9	10	11	12	
3^0	1	–1	0	1	–1	0	1	–1	0	1	–1	0	$\Sigma_1 = 0$
3^1	0	1	1	1	–1	–1	–1	0	0	0	1	1	$\Sigma_2 = 2$
3^2	0	0	0	0	1	1	1	1	1	1	1	1	$\Sigma_3 = 8$

Se observă că nu toate sumele pe linii sunt nule. Se pot nega(tiviza) unele coloane pentru a face suma pe linii nulă. De exemplu, prin schimbarea de semn a coloanelor 7, 9, 11 si 12 se obtine

	1	2	3	4	5	6	7	8	9	10	11	12	
3^0	1	–1	0	1	–1	0	–1	–1	0	1	1	0	$\Sigma_1 = 0$
3^1	0	1	1	1	–1	–1	1	0	0	0	–1	–1	$\Sigma_2 = 0$
3^2	0	0	0	0	1	1	–1	1	–1	1	–1	–1	$\Sigma_3 = 0$

Acum se plasează monedele pe balantă porivit regulii următoare: pentru cântărirea numărul i, se plasează moneda n

- pe talerul stâng dacă $n_i = -1$;
- alături dacă $n_i = 0$;
- pe talerul din dreapta dacă $n_i = 1$.

Rezultatul celor trei cântăriri va stabili moneda diferită (dacă există una) si va spune dacă este mai grea sau mai usoară. Rezultatul fiecărei cântăriri este 0 dacă cele două talere sunt în echilibru, –1 dacă talerul stâng este mai greu, 1 dacă talerul drept este mai greu. Atunci cele trei cântăriri dau dezvoltarea ternară a indicelui monedei cu defect de greutate. Dacă dezvoltarea aste aceeasi cu aceea din matrice, ea indică faptul că moneda este mai grea. Dacă dezvoltarea este de semn contrar, atunci moneda este mai usoară. De exemplu, 0(–1)(–1) conduce la $0\times 3^0 + (-1)\times 3^1 + (-1)\times 3^2 = -12$ ceea ce indică moneda numărul 12 ca fiind mai grea, 10(–1) indică moneda numărul 8 ca fiind mai usoară, 000 indică absenta monedei cu defect.
De ce această schemă lucrează? Ea este un cod Hamming corector de o eroare pentru alfabetul ternar (v.capitolul dedicat subiectului). Urmează câteva detalii.

De observat mai întâi câteva proprietăți ale matricii de mai sus care este folosită în schemă. Toate coloanele sunt distincte si nici o pereche de coloane adunate nu dau vectorul nul. Totodată, dacă o monedă este mai grea, ea va produce secventa de cântăriri care se potriveste cu coloana ei din matrice. Dacă moneda este mai usoară, produce ca secventă de cântăriri negativa coloanei ei. Prin combinarea tuturor acestor fapte, se poate vedea că orice monedă (unică) diferită ca greutate va produce o secventă de cântăriri (unică) si moneda poate fi determinată din acea secventă.
Una din întrebările care se pot formula este dacă limita stabilită la punctul (a) este realmente accesibilă. De pildă, se pot tria 13 monede prin trei cântăriri? Desi nu se poate cu o schemă ca aceea de mai sus, *se poate* în ipoteza în care s-a extras limita de mai devreme. Limita nici nu interzice împărtirea monedelor în două submultimi, nici nu exclude existenta unei alte monede cunoscută ca normală. În oricare din aceste condiții, este posibil a găsi moneda diferită dintre cele 13 prin 3 cântăriri.

Problema 25. Exemplu de entropie combinată

Fie probabilitătile $p(x, y)$ date de tabelul alăturat.

Y / X	0	1
0	1/3	1/3
1	0	1/3

Stabiliti:
a. $H(X)$ si $H(Y)$
b. $H(Y|X)$ si $H(X|Y)$
c. $H(Y, X)$
d. $H(Y) – H(Y|X)$

Solutie:
a. $H(X) = (2/3)\log(3/2) + (1/3)\log 3 \approx 0{,}918$ biti. Aceeasi entropie pentru Y.
b. $H(X|Y) = (1/3)H(X|Y = 0) + (2/3)H(X|Y = 1) \approx 0{,}667$ biti. Rezultat identic si pentru $H(Y|X)$.
c. $H(X,Y) = 3(1/3)\log 3 \approx 1{,}585$ biti.
d. $H(Y) – H(Y|X) \approx 0{,}251$ biti.

Problema 26.

Arătati că dacă $H(Y/X) = 0$ atunci Y este o functie de X, adică pentru orice x cu $p(x) > 0$ există numai o valoare posibilă pentru y cu $p(x, y) > 0$.

Problema 27.

În sporturile de echipă sunt campionate care se încheie cu o serie de sapte jocuri. Seria se încheie când una din echipe câstigă patru jocuri. Fie X variabila aleatoare care reprezintă rezultatul seriei de jocuri dintre echipele A si B; valorile posibile ale lui X sunt AAAA, BABABAB, BBBAAAA etc. Fie Y numărul de jocuri efectiv jucate, număr între 4 si 7. Admitând că echipele A si B sunt egale ca tărie si că jocurile sunt independente unul de altul, să se calculeze $H(X)$, $H(Y)$, $H(Y/X)$ si $H(X/Y)$.

Problema 28.

Fie X si Y două variabile aleatoare care iau valorile $x_1, x_2, \ldots, x_r$, respectiv $y_1, y_2, \ldots, y_s$ si fie $Z = X + Y$.

a. Să se verifice prin calcul direct că $H(Z/X) = H(Y/X)$. Pe baza acestui fapt, argumentati că dacă X si Y sunt independente, atunci $H(Y) \leq H(Z)$. (*Indicatie*: informatia este totdeauna nenegativă)
b. Dati un exemplu în care $H(X) > H(Z)$ si $H(Y) > H(Z)$. (*Indicatie*: încercati să faceti $H(Z) = 0$)
c. În ce conditii are loc relatia $H(Z) = H(X) + H(Y)$?

Problema 29.

Arătati că entropia pentru distributia probabilistică $(p_1, \ldots, p_i, \ldots, p_j, \ldots, p_m)$ este mai redusă decât cea a distributiei $(q_1, \ldots, q_m)$ în care $q_i = q_j = (p_i + p_j)/2$ si $q_k = p_k$ pentru orice k diferit de i si j. (*Indicatie*: calculati $H_p - H_q$ uzând de relatia de definitie). Observati că acest rezultat particular este legat de un adevăr general: împingerea unei distributii spre uniformitate are ca efect cresterea entropiei.
Solutie. Entropia sursei în forma originară este

$$H_p = -\sum_{k=1}^{m} p_k \log p_k$$

Entropia sursei modificate este

$$H_q = -\sum_{k=1}^{m} q_k \log q_k$$

În cele două sume cei mai multi termeni sunt identici si anume aceia de indice k diferit de i si de j. Asadar, diferenta celor două entropii este

$$H_q - H_p = -2\frac{(p_i + p_j)}{2}\log\frac{(p_i + p_j)}{2} + p_i \log p_i + p_j \log p_j =$$

$$= 2\left[-\left(\frac{1}{2}p_i + \frac{1}{2}p_i\right)\log\left(\frac{1}{2}p_i + \frac{1}{2}p_i\right) - \left(-\frac{1}{2}p_i \log p_i - \frac{1}{2}p_j \log p_j\right)\right]$$

Pe baza convexitătii functiei $-x\log x$, rezultă pozitivitatea diferentei. Ceea ce trebuia demonstrat.

Problema 30. Entropie diferentială

Evaluati entropia diferentială $h(X) = -\int f \ln f$ pentru:

a. Densitatea exponentială, $f(x) = \lambda e^{-\lambda x}$, $x \geq 0$
b. Densitatea Laplace, $f(x) = (1/2)\lambda e^{-\lambda|x|}$
c. Suma variabilelor aleatoare independente X_1 si X_2 distribuite normal cu mediile μ_i si dispersiile σ_i^2, $i = 1, 2$.

Problema 31.

Fie $p(x)$ o functie de probabilitate. Dovediti că pentru orice $d \geq 0$

$$\Pr\{p(x) \leq d\} \log \frac{1}{d} \leq H(X)$$

(Inegalitatea lui Markov).

Solutie. Probabilitătile sunt numere pozitive, asadar

$$p(x) \leq d \Rightarrow \quad 1/p(x) \geq 1/d.$$

Expresia entropiei variabilei X este

$$H(X) = \int_{-\infty}^{\infty} p(x) \log \frac{1}{p(x)} dx$$

Relatia aceasta combinată cu inegalitatea de mai devreme conduce la

$$H(X) \geq \log \frac{1}{d} \int_{-\infty}^{\infty} p(x) dx = \log \frac{1}{d}$$

Deoarece orice probabilitate este un număr nenegativ subunitar,

$$\log \frac{1}{d} \geq \Pr\{p(x) \leq d\} \log \frac{1}{d}$$

De unde inegalitatea din enunt.

Problema 32. Entropia unui amestec disjunct

Fie X_1 si X_2 variabile aleatoare discrete extrase conform functiilor de probabilitate $p_1(.)$ si $p_2(.)$ pe multimile $\boldsymbol{X_1} = \{1, 2, \ldots, m\}$ si $\boldsymbol{X_2} = \{m + 1, \ldots, n\}$. Cele două multimi nu se intersectează, după cum usor se observă. Fie

$$X = \begin{cases} X_1 & \text{cu probabilitatea } \alpha \\ X_2 & \text{cu probabilitatea } 1 - \alpha \end{cases}$$

a. Exprimati $H(X)$ în functie de $H(X_1)$, $H(X_2)$ si α.
b. Maximizati în raport cu α pentru a arăta că $2^{H(X)} \leq 2^{H(X_1)} + 2^{H(X_2)}$ si interpretati utilizând notiunea după care $2^{H(X)}$ este dimensiunea efectivă a alfabetului.
c. Fie X_1 si X_2 distribuite uniform pe multimile lor alfabetice. Care este valoarea α care maximizează entropia $H(X)$ asociată.

Solutie:

a. Această problemă poate fi rezolvată prin scrierea relatiei de definitie a entropiei si prin dezvoltarea termenilor diversi pe care aceasta în contine. Dar aici se utilizează algebra legată de entropii pentru o demonstratie mai simplă. Deoarece X1 si X2 au multimi de definitie disjuncte, se poate scrie

$$X = \begin{cases} X_1 & \text{cu probabilitatea} \quad \alpha \\ X_2 & \text{cu probabilitatea} \quad 1-\alpha \end{cases}$$

Se defineste o functie de X

$$\theta = f(X) = \begin{cases} 1 & \text{pentru} \quad X = X_1 \\ 2 & \text{pentru} \quad X = X_2 \end{cases}$$

Apoi, ca în **Problema 21**,

$$H(X) = H(X,f(X)) = H(\theta) + H(X|\theta) =$$
$$= H(\theta) + p(\theta = 1)H(X|\theta = 1) + p(\theta = 2)H(X|\theta = 2) =$$
$$= H(\alpha) + \alpha H(X_1) + (1 - \alpha)H(X_2)$$

cu $H(\alpha) = -\alpha\log\alpha - (1 - \alpha)\log(1 - \alpha)$.

b. Punând $F(\alpha) = H(\alpha) + \alpha H(X_1) + (1 - \alpha)H(X_2)$, se observă că F este o functie concavă de a asa încât ea este maximă în punctul de anulare a derivatei. Rezolvând

$$F'(\alpha) = -\log\alpha + \log(1 - \alpha) + H(X_1) - H(X_2) = 0$$

se obtine succesiv

$$\alpha^* = \frac{2^{H(X_1)}}{2^{H(X_1)} + 2^{H(X_2)}}$$

$$F(\alpha^*) = \log\left(2^{H(X_1)} + 2^{H(X_2)}\right).$$

Asadar,

$$H(X) = H(\alpha) + \alpha H(X_1) + (1 - \alpha)H(X_2) = F(\alpha) \le F(\alpha^*) = \log\left(2^{H(X_1)} + 2^{H(X_2)}\right)$$

În final,

$$2^{H(X)} \le 2^{H(X_1)} + 2^{H(X_2)}$$

c. Deoarece X_1 si X_2 sunt uniforme pe multimile lor alfabetice, entropiile lor sunt date de cardinalele acestor multimi. Astfel, $H(X_1) = \log|X_1| = \log m$ si $H(X_2) = \log|X_2| = \log(n - m)$. Prin înlocuirea acestor valori în expresia lui α^* de la punctul anterior, se obtine

$$\alpha^* = \frac{2^{H(X_1)}}{2^{H(X_1)} + 2^{H(X_2)}} = \frac{2^{\log m}}{2^{\log m} + 2^{\log(n-m)}} = \frac{m}{m + (n - m)} = \frac{m}{n}$$

Problema 33. Entropie maximă

Aflati densitatea de maximă entropie f care satisface relatiile $EX = \alpha_1$, $E\ln x = \alpha_2$. În altă formulare, stabiliti

$$\max\{h(f)\}$$

în conditiile

$$\int xf(x)dx = \alpha_1, \int (\ln x)f(x)dx = \alpha_2$$

Ce familie de densităti de probabilitate este aceasta?

Solutie. Distributia de maximă entropie supusă la restrictiile din enunt este de forma

$$f(x) = e^{\lambda_0 + \lambda_1 x + \lambda_2 \ln x} = c x^{\lambda_2} e^{\lambda_1 x}$$

care face parte din familia distributiilor Gamma. Constantele se aleg pentru a satisface relatiile restrictive.

Problema 34. Random walk într-un cub

O pasăre zboară din încăpere în încăpere într-un cub 3×3×3 (egal probabil prin fiecare perete interior). Care este rata entropiei?

Solutie: Rata entropiei unei deplasări aleatoare (random walk) într-un graf este dată de relatia

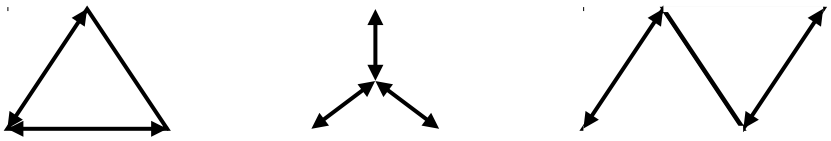
$$H(X) = \log(2E) - H\left(\frac{E_1}{2E}, \ldots, \frac{E_m}{2E}\right)$$

Un cub are 8 vârfuri, 12 muchii, 6 fete si un centru. Vârfurile au 3 arce, muchiile au 4 arce, fetele au 5 arce si centrele au 6 arce. Asadar numărul total de arce este $E = 54$. Astfel,

$$H(X) = \log 108 + 8\left(\frac{3}{108}\log\frac{3}{108}\right) + 12\left(\frac{4}{108}\log\frac{4}{108}\right) +$$
$$+ 6\left(\frac{5}{108}\log\frac{5}{108}\right) + 1\left(\frac{6}{108}\log\frac{6}{108}\right) = 2{,}03 \text{ biti.}$$

Problema 35. Entropia grafurilor

Se consideră o deplasare aleatoare (*random walk*) într-un graf (conex) cu 3 arce.



Graful 1 Graful 2 Graful 3

Grafuri cu trei arce

a. Care graf are cea mai scăzută rată a entropiei? Care este acea rată?
b. Care graf are cea mai înaltă rată a entropiei?

Solutie: Sunt posibile trei grafuri cu trei arce, ca în figura alăturată.
Rata entropiei este dată de

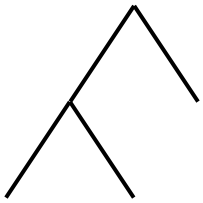
$$H = -\sum_i \mu_i \sum_j p_{ij} \log p_{ij} = \sum_i \frac{w_i}{w} \log w_i$$

Pentru graful 1, $\{w_i\} = \{2, 2, 2\}$ de unde rezultă $H = 3[(2/6)\log 2] = 1$.
Pentru graful 2, $\{w_i\} = \{1, 1, 1, 3\}$ de unde rezultă $H = (3/6)\log 3 \approx 0{,}79$.
Pentru graful 3, $\{w_i\} = \{1, 2, 2, 1\}$ de unde rezultă $H = 2[(2/6)\log 2] \approx 0{,}667$.
Asadar, graful 1 are cea mai mare entropie, graful 3 are cea mai mică entropie.

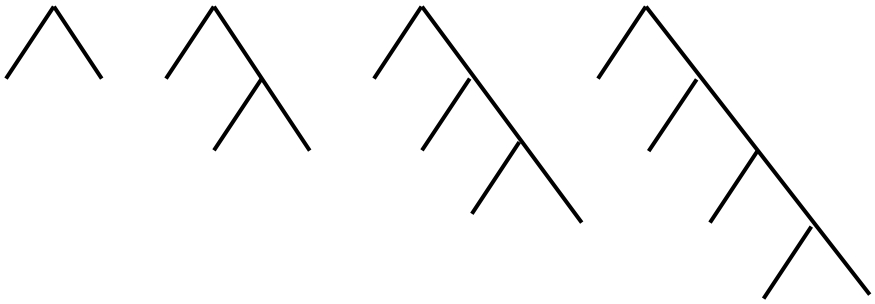
Problema 36. Entropia unui arbore aleator

Se consideră următoarea metodă de generare aleatoare a unui arbore binar cu n noduri. Mai întâi se dezvoltă nodul rădăcină:

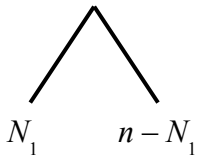
Apoi se dezvoltă unul din cele două noduri terminale la întâmplare:



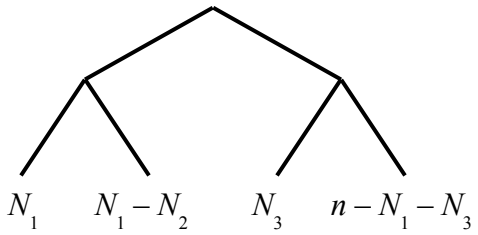
La pasul k, se alege unul din cele $k - 1$ noduri terminale potrivit unei distributii uniforme si se dezvoltă acel nod. Se continuă până când se generează n noduri terminale. Astfel, o secventă care conduce la cinci noduri poate arăta astfel:



Surprinzător, metoda următoare de generare de arbori aleatori produce aceeasi distributie probabilistică pe arbori cu n noduri terminale. Mai întâi se alege un întreg N_1 uniform distribuit pe multimea $\{1, 2, \ldots, n\}$. Se obtine imaginea:



Apoi se alege un întreg N_2 uniform distribuit pe $\{1, 2, \ldots, N_1 - 1\}$ si, independnent un alt întreg N_3 uniform distribuit peste $\{1, 2, \ldots, (n - N_1) - 1\}$. Figura se modifică astfel:



Se continuă procesul până când nu se mai pot face alte subdiviziuni. (Echivalenta celor două scheme de generare decurge, de pildă, din modelul urnei datorat lui Polya).

Se notează acum cu T_n un arbore aleator cu n noduri generat ca mai sus. Distributia probabilistică a acestor arbori pare dificil de descris, dar se poate afla entropia acestei distributii în formă recursivă.

Mai întâi câteva exemple.

Pentru $n = 2$, există un singur arbore. Asadar, $H(T_2) = 0$.

Pentru $n = 3$, sunt doi arbori posibili, egal probabili: entropia este $H(T_3) = \log 2$.

Pentru $n = 4$, sunt posibili cinci arbori, cu probabilitătile 1/3, 1/6, 1/6, 1/6, 1/6.

Acum, pentru o relatie de recurentă, fie $N_1(T_n)$ numărul de noduri terminale ale arborelui T_n în jumătatea din dreapta a acelui arbore. Justificati fiecare din pasii următori:

$$\begin{aligned} H(T_n) &= H(N_1, T_n) \\ &= H(N_1) + H(T_n \mid N_1) \\ &= \log(n-1) + H(T_n \mid N_1) \\ &= \log(n-1) + \frac{1}{n-1}\sum_{k=1}^{n-1} [H(T_k) + H(T_{n-k})] \\ &= \log(n-1) + \frac{2}{n-1}\sum_{k=1}^{n-1} H(T_k) \end{aligned}$$

Utilizati succesiunea de egalităti de mai sus pentru a arăta că

$$(n-1)H_n = nH_{n-1} + (n-1)\log(n-1) - (n-2)\log(n-2)$$

sau

$$\frac{H_n}{n} = \frac{H_{n-1}}{n-1} + c_n$$

cu adaosul c_n potrivit definit. Deoarece $\Sigma c_n = c < \infty$, s-a dovedit prin aceasta că $H(T_n)/n$ tinde către o constantă. Astfel, numărul mediu statistic de biti necesari pentru a descrie arborele aleator T_n creste liniar cu n.

Solutie:

Prin conditionarea în lant a entropiilor si pentru că N_1 este o functie de T_n, $H(T_n,N_1) = H(T_n) + H(N_1|T_n) = H(T_n) + 0$.

Prin conditionarea în lant a entropiilor, $H(T_n,N_1) = H(N_1) + H(T_n|N_1)$.

$H(N_1) = \log(n-1)$ deoarece N_1 este uniform pe $\{1, 2, \ldots, n-1\}$.

Prin definitia entropiei conditionate,

$$H(T_n \mid N_1) = \sum_{k=1}^{n-1} p(N_1 = k) H(T_n \mid N_1 = k) = \frac{1}{n-1} \sum_{k=1}^{n-1} H(T_n \mid N_1 = k)$$

Deoarece conditionat de N_1, subarborele din stânga si subarborele din dreapta sunt alegeri independente, $H(T_n|N_1 = k) = H(T_k, T_{n-k}|N_1 = k) = H(T_k) + H(T_{n-k})$ asa încât

$$H(T_n \mid N_1) = \frac{1}{n-1} \sum_{k=1}^{n-1} [H(T_k) + H(T_{n-k})]$$

Printr-o simplă schimbare de variabile,

$$\sum_{k=1}^{n-1} H(T_{n-k}) = \sum_{k=1}^{n-1} H(T_k)$$

Dacă punem $H_n = H(T_n)$,

$$(n-1)H_n = (n-1)\log(n-1) + 2\sum_{k=1}^{n-1} H_k$$

$$(n-2)H_{n-1} = (n-2)\log(n-2) + 2\sum_{k=1}^{n-2} H_k$$

Prin scăderea relatiei din urmă din cea anterioară se obtine

$$(n-1)H_n - (n-2)H_{n-1} = (n-1)\log(n-1) - (n-2)\log(n-2) + 2H_{n-1}$$

sau

$$\frac{H_n}{n} = \frac{H_{n-1}}{n-1} + \frac{\log(n-1)}{n} - \frac{(n-2)\log(n-2)}{n(n-1)} = \frac{H_{n-1}}{n-1} + C_n$$

Asupra termenului C_n se mai poate lucra:

$$C_n = \frac{\log(n-1)}{n} - \frac{(n-2)\log(n-2)}{n(n-1)} = \frac{\log(n-1)}{n} - \frac{\log(n-2)}{n-1} + \frac{2\log(n-2)}{n(n-1)}$$

Substituind ecuatia pentru H_{n-1} în ecuatia pentru H_n si procedând recursiv, se obtine o sumă telescopică

$$\frac{H_n}{n} = \sum_{i=1}^{n} C_i + \frac{H_2}{2} = \sum_{i=3}^{n} \frac{2\log(i-2)}{i(i-1)} + \frac{1}{n}\log(n-1)$$

Deoarece limita când n $\to \infty$ a ultimului termen din expresia din urmă este nulă

$$\lim_{n\to\infty} \frac{H_n}{n} = \sum_{i=3}^{\infty} \frac{2\log(i-2)}{i(i-1)} \le \sum_{i=3}^{\infty} \frac{2\log(i-1)}{(i-1)^2} = \sum_{i=2}^{\infty} \frac{2}{i^2}\log i$$

Pentru i suficient de mare, $\log i \le \sqrt{i}$ de unde rezultă că ultima serie este convergentă (este majorată de seria cu termenul general $i^{-(3/2)}$, convergentă la rândul ei).

Limita de mai devreme există. O evaluare pe calculator arată că ea are valoare aproximativă de 1,736 biti.

Asadar, numărul de biti necesari pentru a descrie un arbore binar aleator cu n noduri creste liniar cu n.

Problema 37. Lanț Markov

Se dă matricea

$$P = [P_{ij}] = \begin{bmatrix} 1/2 & 1/4 & 1/4 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/4 & 1/2 \end{bmatrix}$$

Fie X_1 distribuit uniform pe stările {0, 1, 2}. Fie $\{X_i\}_1^\infty$ un lanț Markov cu matricea de tranzitie P, astfel încât $\Pr(X_{n+1} = j | X_n = i) = P_{ij}$, $i, j \in \{0, 1, 2\}$.

a. Este $\{X_n\}$ stationar?
b. Aflati $\lim_{n\to\infty} H(X_1, \ldots, X_n)$.

Considerati acum procesul derivat $Z_1, Z_2, \ldots, Z_n$ în care

$$Z_1 = X_1$$
$$Z_i = X_i - X_{i-1} \pmod 3, \quad i = 2, 3, \ldots, n$$

Asadar, Z^n face o codare a tranzitiilor, nu a stărilor.

c. Aflati $H(Z_1, Z_2, \ldots, Z_n)$.
d. Aflati $H(Z_n)$ si $H(X_n)$ pentru $n \geq 2$.
e. Aflati $H(Z_n|Z_{n-1})$ pentru $n \geq 2$.
f. Sunt, pentru $n \geq 2$, Z_{n-1} si Z_n independente?

Solutie:

a. Fie μ_n functia de probabilitate la momentul n. Deoarece $\mu_1 = (1/3, 1/3, 1/3)$ si $\mu_2 = \mu_1 P = \mu_1$, $\mu_n = \mu_1 = (1/3, 1/3, 1/3)$ pentru orice n si $\{X_n\}$ este stationar.
   Alternativ, observatia că P este dublu stochastică conduce la aceeasi concluzie.
b. Deoarece $\{X_n\}$ este un proces Markov stationar

$$\lim_{n\to\infty} H(X_1,\ldots,X_n) = H(X_2 \mid X_1) = \sum_{k=0}^{2} P(X_1 = k) H(X_2 \mid X_1 = k) =$$
$$= 3 \cdot \frac{1}{3} \cdot H\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right) = \frac{3}{2}$$

c. Deoarece secventele $(X_1, \ldots, X_n)$ si $(Z_1, \ldots, Z_n)$ sunt în corespondentă biunivocă, prin regula conditionărilor succesive ale entropiilor si din markovianism

$$H(Z_1, \ldots, Z_n) = H(X_1, \ldots, X_n) = \sum_{k=1}^{n} H(X_k \mid X_1,\ldots,X_{k-1}) =$$
$$= H(X_1) + \sum_{k=2}^{n} H(X_k \mid X_{k-1}) = H(X_1) + (n-1) H(X_2 \mid X_1) = \log 3 + \frac{3}{2}(n-1)$$

   Ca alternativă, se pot utiliza rezultatele de la punctele următoare, (d), (e) si (f). Deoarece $Z_1, \ldots, Z_n$ sunt independente si $Z_2, \ldots, Z_n$ sunt distribuite identic, cu functia de probabilitate (1/2, 1/4, 1/4),

$$H(Z_1, \ldots, Z_n) = H(Z_1) + \ldots + H(Z_n) = H(Z_1) + (n-1)H(Z_2) = \log 3 + \frac{3}{2}(n-1)$$

d. Deoarece $\{X_n\}$ este un proces Markov stationar cu $\mu_n = (1/3, 1/3, 1/3)$,
   $$H(X_n) = H(X_1) = H(1/3, 1/3, 1/3) = \log 3.$$
   Pentru $n \geq 2$

$$Z_n = \begin{cases} 0 & \text{cu probabiltatea } 1/2 \\ 1 & \text{cu probabiltatea } 1/4 \\ 2 & \text{cu probabiltatea } 1/4 \end{cases}$$

Asadar,

$$H(Z_n) = H(1/2, 1/4, 1/4) = 3/2.$$

e. Datorită simetriei lui P, $\Pr[Z_n|Z_{n-1}] = \Pr[Z_n]$ pentru orice $n \geq 2$. Astfel, $H[Z_n|Z_{n-1}] = H[Z_n] = 3/2$.
Ca demonstratie alternativă, utilizând rezultatul de la punctul (f), se poate ajunge elementar la aceeasi concluzie.

f. Fie $k \geq 2$. Se observă din simetria lui P că $Z_{k+1} = X_{k+1} - X_k$ este independent de X_k. Acum din

$$\Pr[Z_{k+1}|X_k, X_{k-1}] = \Pr[X_{k+1} - X_k|X_k, X_{k-1}] = \Pr[X_{k+1} - X_k|X_k] =$$
$$= \Pr[X_{k+1} - X_k] = \Pr[Z_{k+1}]$$

rezultă că Z_{k+1} este independent de (X_k, X_{k-1}) si este deci independent si de $Z_k = X_k - X_{k-1}$. Pentru $k = 1$, din nou din simetria lui P rezultă imediat că Z_2 este independent de $Z_1 = X_1$.

Problema 38. Markov de ordinul doi

Fie $\{X_n\}$, $X_n \in \{0, 1\}$ un proces stochastic binar. Fie $X_{n+1} = X_n \oplus X_{n-1} \oplus Z_{n+1}$, cu $\oplus$ operatorul de adunare modulo 2 si $\{Z_n\}$ un proces Bernoulli(p). Care este rata entropiei pentru $\{X_n\}$? Ar putea fi util a redefini acest proces Markov de ordinul doi ca un proces Markov de primul ordin.

Solutie: Fie $Y_i = (X_i, X_{i-1})$ pentru orice i. Este usor de văzut că $\lim_{n\to\infty} H(X^n)/n = \lim_{n\to\infty} H(Y^n)/n$, si că $\{Y_i\}$ este un proces Markov de ordinul 1. Asadar rata entropiei pentru $\{X_n\}$ este aceeasi cu rata entropiei procesului $\{Y_n\}$. Deoarece $\{Y_n\}$ este un proces Markov de ordinul întâi, are loc

$$\lim_{n\to\infty} \frac{1}{n} H(Y_n) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H(Y_i \mid Y_{i-1}) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H(p) = H(p)$$

Se poate conchide că $\{X_n\}$ are rata entropiei $H(p)$.

Alternativ, se poate ataca direct rata entropiei unui proces Markov de ordinul 2 $\{X_n\}$ astfel

$$\lim_{n\to\infty} \frac{1}{n} H(X^n) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H(X_i \mid X^{i-1}) =$$

$$= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H(X_i \mid X_{i-1}, X_{i-2}) \to H(p)$$

în ultima egalitate se evidentiază ordinul procesului, 2, iar limita provine din faptul că $H(X_n \mid X_{n-1}, X_{n-2}) = H(p)$ pentru orice $n \geq 3$ (Cesáro).

CANALE DE TRANSMITERE A INFORMATIEI

Lucrarea 4

Tema: Transinformatia si capacitatea canalelor.

Canalul are un alfabet de intrare alcătuit din *n* simboluri. Alfabetul observat la iesire este compus din *m* simboluri.

Matricea de probabilităti conditionate care caracterizează canalul este o matrice $n \times m$ cu sumele elementelor fiecărei linii egale cu unitatea. În program, această matrice este generată aleator.
Sursa de informatie care debitează la intrarea canalului este caracterizată de un vector (linie) cu n elemente, probabilitătile de aparitie ale simbolurilor sursei. Vectorul acesta este generat, de asemenea, aleator. Se evaluează o entropie a sursei numită si entropie *apriori*.
Probabilitătile simbolurilor sursei observate la iesirea canalului rezultă prin multiplicarea vectorului asociat sursei cu matricea canalului. Rezultatul acestei operatii este un vector cu m componente care însumate dau tot unitatea. Entropia acestei surse secundare este uzual diferită de entropia apriori datorită "lucrării" canalului asupra informatiei generate de sursă.
În conditiile observării unui anumit simbol/caracter la iesire se pot evalua probabilităti conditionate de acest eveniment pentru fiecare simbol al sursei. Este ceea ce se face în program în secventa mediană, acolo unde se calculează matricea `pab`. Coloanele acestei matrici sunt tocmai probabilitătile conditionate mentionate. Suma lor este unitară si se calculează o entropie *aposteriori* a sursei în fiecare caz, pentru fiecare din cele m observatii posibile la iesirea canalului.
O medie a acestor entropii care ia în considerare probabilitătile simbolurilor la iesire produce o entropie a sursei de la intrare conditionată de sursa de la iesirea canalului.
Diferenta dintre entropia (apriori) a sursei si entropia conditionată evaluată astfel poartă numele de *transinformatie* sau de *informatie mutuală* si este cantitatea de informatie transmisă prin canal.
Transinformatia este variabilă pentru surse diferite care debitează la intrarea canalului. Variabilitatea se referă numai la lista probabilitătilor de aparitie a simbolurilor sursei si nu la modificarea vreuneia din cele două multimi alfabetice, de intare si de iesire, cu care lucrează canalul.
Transinformatia este o mărime simetrică: dacă intrarea devine iesire si viceversa, dacă matricile de probabilităti conditionate se schimbă între ele, `pab` în loc `p`, se obtine aceeasi valoare a transinformatiei. Un canal lucrează la fel de bine (sau de rău) si într-un sens si în celălalt. Se propune studentilor realizarea unei variante a programului capabilă a evalua informatia mutuală prin canal în situatia aceasta modificată.
Există o limită superioară a transinformatiei, care nu poate fi depăsită. Această transinformatie maximă poartă numele de *capacitate a canalului*. Capacitatea canalului se poate evalua analitic rezolvând o problemă de extrem supus la legături/restrictii, pe o multime de n variabile, probabilitătile simbolurilor de la intare, supuse la legătura/restrictia unică de sumă generală egală cu unitatea.
În program s-a preferat numai o explorare a limitei maxime pentru transinformatie prin generarea aleatoare a mai multor surse cu n simboluri si prin evaluarea de fiecare dată a informatiei mutuale. Se poate observa o cea-mai-mare-valoare a transinformatiei între aceste "realizări" posibile, care se apropie de capacitatea canalului.

Se sugerează si explorarea canalelor binare ($n = m = 2$) simetrice (matricea **p** simetrică fată de diagonala principală) si a canalelor binare cu stergere ($n = 2$, $m = 3$) care au în alfabetul de iesire, înafară de valorile binare obisnuite si un al treilea simbol care nu poate fi asimilat niciunuia din valorile transmise.
Se propune următorul script Matlab:

```
clear
n=3;
m=4;
p=rand(n,m);
for i=1:n
    suml=sum(p(i,:));
    p(i,:)=p(i,:)/suml;
end    % crearea unei matrici a canalului
matricea_canalului=p
a=rand(1,n);
a=a/sum(a);    % probabilitati sursa de intrare
b=a*p;    % probabilitati sursa observata la iesire
probabilitatile_simbolurilor_de_intrare=a
ha=entropie(a);    % entropia apriori
entropia_apriori=ha
probabilitatile_simbolurilor_de_iesire=b
hb=entropie(b);
entropia_sursei_observate_la_ieisre=hb
pab=zeros(n,m);
for i=1:n
    for j=1:m
        pab(i,j)=p(i,j)*a(i)/b(j);
    end
end
probabilitati_intrare_condiditionate_de_iesiri=pab
for j=1:m
    enap(j)=entropie((pab(:,j))');
end    % entropii aposteriori
entropii_aposteriori=enap
entropia_conditionata_de_iesirea_observata=b*enap'
transinformatia=ha-b*enap'
display('*** Uremaza o cautare aleatoare a capacitatii canalului
***')
display('*** Pentru continuare apasati Enter! ***')
pause    % cautarea aleatoare a capacitatii canalului
for k=1:20
    a=rand(1,n);a=a/sum(a);b=a*p;
    ha=entropie(a);
    for i=1:n
        for j=1:m
            pab(i,j)=p(i,j)*a(i)/b(j);
        end
    end
    for j=1:m
        enap(j)=entropie((pab(:,j))');
    end
    sursa_la_intrare_si_transinformatia=[a ha-b*enap']
```

```
end
```

Pentru un canal binar simetric, secventa introductivă a scriptului se modifică astfel:

```
clear
n=2;
m=2;
p=rand(n,m);
for i=1:n
    suml=sum(p(i,:));
    p(i,:)=p(i,:)/suml;
end    % crearea unei matrici a canalului
p(2,1)=p(1,2);p(2,2)=p(1,1);    % simtetrizarea
```

Modificările aduse aceleiasi secvente pentru cazul unui canal binar simetric cu stergere sunt următoarele:

```
clear
n=2;
m=3;
p=rand(n,m);
for i=1:n
    suml=sum(p(i,:));
    p(i,:)=p(i,:)/suml;
end    % crearea unei matrici a canalului
p(2,1)=p(1,3);p(2,2)=p(1,2);p(2,3)=p(1,1);    % simterizarea
```

Problema 39.

Arătati că un canal de transmitere a informatiei caracterizat de matricea

$$P = \begin{bmatrix} 2/3 & 1/3 & 0 \\ 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 2/3 \end{bmatrix}$$

are o capacitate care este atinsă la o intrare care are un simbol de probabilitate nulă. Care este capacitatea acestui canal? Găsiti o explicatie intuitivă a faptului că una din literele alfabetului de intrare nu este folosită.

Solutie: Fie probabilitătile simbolurilor de la intrare $P(A) = [p \quad q \quad r]$ cu $p + q + r = 1$. Atunci, probabilitătile simbolurilor de iesire sunt

$$P(B) = [p \quad q \quad r]\begin{bmatrix} 2/3 & 1/3 & 0 \\ 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 2/3 \end{bmatrix} = \begin{bmatrix} \frac{2}{3}p + \frac{1}{3}q & \frac{1}{3} & \frac{1}{3}q + \frac{2}{3}r \end{bmatrix}$$

si probabilitătile simbolurilor de la intrare (A), conditionate de cele de la iesire (B) sunt

$$P(A/B) = \begin{bmatrix} 2p/(2p+q) & q/(2p+q) & 0 \\ p & q & r \\ 0 & q/(q+2r) & 2r/(q+2r) \end{bmatrix}$$

Entropia apriori a sursei A este

$$H(A) = p\log\frac{1}{p} + q\log\frac{1}{q} + r\log\frac{1}{r}$$

Entropiile aposteriori ale sursei A sunt

$$H(A/b_1) = \frac{2p}{2p+q}\log\frac{2p+q}{2p} + \frac{q}{2p+q}\log\frac{2p+q}{q}$$

$$H(A/b_2) = p\log\frac{1}{p} + q\log\frac{1}{q} + r\log\frac{1}{r} = H(A)$$

$$H(A/b_3) = \frac{q}{q+2r}\log\frac{q+2r}{q} + \frac{2r}{q+2r}\log\frac{q+2r}{2r}$$

si entropia sursei A conditionată de B este

$$H(A/B) = \frac{2p+q}{3}H(A/b_1) + \frac{1}{3}H(A/b_2) + \frac{q+2r}{3}H(A/b_3) =$$

$$= \frac{2}{3}p\log\frac{1}{2p} + \frac{1}{3}q\log\frac{1}{q} + \frac{1}{3}(2p+q)\log(2p+q) + \frac{1}{3}H(A) +$$

$$+ \frac{1}{3}q\log\frac{1}{q} + \frac{2}{3}r\log\frac{1}{2r} + \frac{1}{3}(q+2r)\log(q+2r) =$$

$$= \frac{2}{3}H(A) - \frac{2}{3}(p+r) + \frac{1}{3}H(A) +$$

$$+ \frac{1}{3}(2p+q)\log(2p+q) + \frac{1}{3}(q+2r)\log(q+2r) =$$

$$= H(A) - \frac{2}{3}(p+r) - \frac{2}{3}\frac{2p+q}{2}\log\frac{2}{2p+q} - \frac{2}{3}\frac{q+2r}{2}\log\frac{2}{q+2r} + \frac{2}{3}$$

Transinformatia (informatia mutuală) care trebuie maximizată pentru a stabili capacitatea canalului este dată de relatia

$$I(A;B) = H(A) - H(A/B) =$$

$$= \frac{2}{3}(p+r) + \frac{2}{3}\left(\frac{2p+q}{2}\log\frac{2}{2p+q} + \frac{q+2r}{2}\log\frac{2}{q+2r}\right) - \frac{2}{3}$$

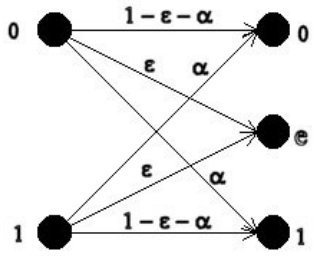
Paranteza contine expresia entropiei unei surse cu două simboluri. Aceasta este maximă când simbolurile sunt echiprobabile

$$\frac{2p+q}{2} = \frac{q+2r}{2}$$

adică atunci când $p = r$. Primul termen este maxim când $p = r = 1/2$ si $q = 0$. Ultimul termen este constant si nu contează la maximizarea transinformatiei.

Problema 40.

Se consideră un canal cu intrări binare care are si erori si stergere (stergere = aparitia la iesire a unui al treilea simbol, neuzual). Fie α probabilitatea erorii si fie ε probabilitatea stergerii astfel încât canalul este reprezentat de graful din figură:



Aflati capacitatea acestui canal.
Particularizati pentru cazul canalului binar simetric ($\varepsilon = 0$).
Particularizati pentru cazul canalului binar cu stergere ($\alpha = 0$).

Problema 41.

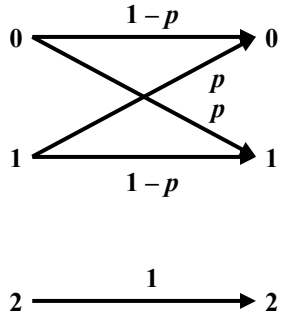
Se consideră un canal cu alfabet binar care preia simboluri de doi biti si produce la iesire simboluri de doi biti conform asocierii următoare: 00 → 01, 01 → 10, 10 → 11 si 11 → 00. Astfel, dacă secventa de doi biti 01 este introdusă în canal, iesirea este 10 cu probabilitatea 1. Fie X_1, X_2 două simboluri la intrare si Y_1, Y_2 simbolurile de iesire corespunzătoare.
Calculati informatia mutuală $I(X_1, X_2 ; Y_1, Y_2)$ ca functie de distributia la intrare a celor patru perechi de intrare posibile.
Arătati că pentru transmiterea în perechi capacitatea acestui canal este de 2 biti.
Arătati că la maximizare prin distributia la intrare, $I(X_1 ; Y_1) = 0$.
Asadar, distributia secventelor de intrare care atinge capacitatea canalului nu maximizează în mod necesar informatia mutuală între simboluri individuale si corespunzătoarele lor la iesire.

Problema 42. Canal sumă

Se consideră două canale fără memorie $(X_1, p(y_1/x_1), Y_1)$ si $(X_2, p(y_2/x_2), Y_2)$ cu capacitătile C_1, respectiv C_2. Se defineste un canal nou astfel: alfabetul de intrare este reuniunea celor două multimi alfabetice X_1 si X_2, alfabetul de iesire este reuniunea celor două multimi alfabetice Y_1 si Y_2 si matricea de probabilităti de tranzitie este $p(y/x) = p(y_1/x_1)$ dacă $x \in X_1$ si $p(y/x) = p(y_2/x_2)$ dacă $x \in X_2$. Asadar, canalul are la dispozitie ambele canale simple, dar acestea sunt utilizate alternativ. Arătati că acest canal sumă are capacitatea

$$C = \log(2^{C_1} + 2^{C_2})$$

Utilizati acest rezultat pentru a calcula capacitatea canalului următor:



Problema 43. Capacitatea canalelor

Calculati capacitatea canalelor de mai jos, caracterizate de multimile alfabetice specificate si de matricile de tranzitie alăturate:

$$X = Y = \{0, 1, 2\},\ p(y \mid x) = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$X = Y = \{0, 1, 2\},\ p(y \mid x) = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \end{bmatrix}$$

$$X = Y = \{0, 1, 2, 3\},\ p(y \mid x) = \begin{bmatrix} p & 1-p & 0 & 0 \\ 1-p & p & 0 & 0 \\ 0 & 0 & q & 1-q \\ 0 & 0 & 1-q & q \end{bmatrix}$$

Problema 44. Capacitatea canalelor

Care sunt capacitătile canalelor

$$X\begin{Bmatrix} 1 & p & 1-p & 0 & 0 & 1 \\ 2 & 0 & p & 1-p & 0 & 2 \\ 3 & 0 & 0 & p & 1-p & 3 \\ 4 & 1-p & 0 & 0 & p & 4 \end{Bmatrix} Y$$

$$X\begin{Bmatrix} 1 & p & 1-p & 0 & 1 \\ 2 & 0 & p & 1-p & 2 \\ 3 & 1-p & 0 & p & 3 \end{Bmatrix} Y\ ?$$

Problema 45. Canal cu stergere

Fie $\{X, p(y|x), Y\}$ un canal discret fără memorie de capacitate C. Se admite că acest canal este conectat imediat în cascadă cu un canal cu stergere $\{Y, p(s|y), S\}$ care sterge fractia α din simboluri.

Mai precis, $S = \{y_1, y_2, \ldots, y_m, e\}$ si $\Pr[S = y|X = x] = (1 - \alpha)p(y|x)$ pentru $y \in Y$ si $\Pr[S = e|X = x] = \alpha$.
Evaluati capacitatea acestui canal.

Problema 46. Informatia mutuală între fetele unei monede

a. Se consideră o aruncare a unei monede corecte. Care este informatia mutuală între fata de deasupra si fata de dedesubt a monedei?
b. Un zar corect cu 6 fete este făcut să se rostogolească. Care este informatia mutuală între fata de deasupra si fata de dedesubt?
c. Care este informatia mutuală între fața de deasupra a unui zar si fața din față (cea mai vizibilă pentru experimentator)?

Solutie:

a. Se observă că
$$I(T; B) = H(B) - H(B|T) = \log 2 = 1$$
deoarece $B \sim$ Bernoulli(1/2) si B este o functie de T. Aici B si T sunt prescurtări pentru "bottom" si "top".
b. Se observă că fata de dedesubt B este din nou functie de fata de deasupra T ($B + T = 7$) si sunt sase posibilități egal probabile pentru B. Asadar,
$$I(T; B) = H(B) - H(B|T) = \log 6 = \log 3 + 1$$
c. Se observă că având rezultatul relativ la fața din față F, rămân patru posibilități egal probabile pentru fața de deasupra T. Asadar,
$$I(T; F) = H(B) - H(B|F) = \log 6 - \log 4 = \log 3 - 1$$
deoarece T este repatizat uniform pe $\{1, 2, \ldots, 6\}$.

Problema 47. Dubla privire

Comparati informatia mutuală $I(X; Y_1, Y_2)$ pe care (Y_1, Y_2) o furnizează despre X, cu suma informatiilor mutuale $I(X; Y_1)$ si $I(X; Y_2)$ pentru fiecare din cele două functii de probabilitate de mai departe.

a. Două priviri independente:
$$p(x, y_1, y_2) = p(x)p(y_1|x)p(y_2|x)$$
b. O privire la două observatii independente:
$$p(x, y_1, y_2) = p(y_1)p(y_2)p(x|y_1, y_2)$$

Solutie:

a. Două priviri independente:
$$\begin{aligned} I(X; Y_1, Y_2) &= H(Y_1, Y_2) - H(Y_1, Y_2|X) = \\ &= [H(Y_1) + H(Y_2) - I(Y_1; Y_2)] - H(Y_1, Y_2|X) = \\ &= [H(Y_1) + H(Y_2) - I(Y_1; Y_2)] - [H(Y_1|X) + H(Y_2|X)] = \\ &= I(X; Y_1) + I(X; Y_2) - I(Y_1; Y_2) \leq I(X; Y_1) + I(X; Y_2) \end{aligned}$$

În succesiunea de relatii de mai sus s-au utilizat: relatia de definitie a informatiei mutuale, formula pentru calculul entropiei surselor compuse, absenta conditionării între Y_1 si Y_2 când este dat X si nenegativitatea informatiei mutuale.

b. O privire la două observatii independente:

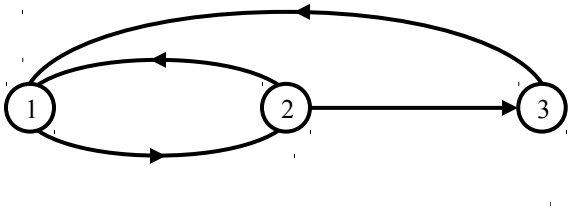
$$I(X;\, Y_1,\, Y_2) = H(Y_1,\, Y_2) - H(Y_1,Y_2|X) =$$
$$= [H(Y_1) + H(Y_2)] - H(Y_1,Y_2|X) =$$
$$= [H(Y_1) + H(Y_2)] - [H(Y_1|X) + H(Y_2|X) - I(Y_1;Y_2|X)] =$$
$$= I(X;Y_1) + I(X;Y_2) + I(Y_1;Y_2|X) \geq I(X;Y_1) + I(X;Y_2)$$

Aici s-au folosit: relatia de definitie a informatiei mutuale, independenta între Y_1 si Y_2, identitatea pentru entropia $H(Y_1,Y_2|X)$, din nou relatia de definitie a informatiei mutuale si nenegativitatea informatiei mutuale.

Problema 48. Rata entropiei pentru secvente restrictionate

În cazul înregistrărilor magnetice, mecanismul de înregistrare si de redare a bitilor impune restrictii asupra secventei de biti care poate fi înregistrată. De pildă, pentru a asigura sincronizarea adecvată, este necesar adesea a limita lungimea secventelor de zerouri între doi de unu. De asemenea pentru a reduce interferenta între simboluri, poate fi impus ca necesar cel putin un zero între doi de unu. Se arată în continuare un exemplu simplu de astfel de restrictii.
Se presupune că se cere ca într-o secventă, între doi de 1 să fie cel putin un 0 si cel mult doi de zero. Astfel, secvente de genul 101001 si 0101001 sunt valide, dar 0110010 si 0000101 nu sunt. Se urmăreste calculul numărului de secvente valide de lungime n.

a. Arătati că multimea de secvente restrictionate este aceeasi cu multimea căilor permise în diagrama de stare alăturată.



b. Fie $X_i(n)$ numărul de căi valide de lungime n cu finalul în starea i. Argumentati că $\mathbf{X}(n) = [X_1(n)\ X_2(n)\ X_3(n)]^T$ verifică relatia de recurentă de mai jos:

$$\begin{bmatrix} X_1(n) \\ X_2(n) \\ X_3(n) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1(n-1) \\ X_2(n-1) \\ X_3(n-1) \end{bmatrix} = A\mathbf{X}(n-1)$$

cu conditia initială $\mathbf{X}(1) = [1\ \ 1\ \ 0]^T$.

c. Apoi, prin inductie are loc

$$\mathbf{X}(n) = A\mathbf{X}(n-1) = A^2\mathbf{X}(n-2) = \ldots = A^{n-1}\mathbf{X}(1)$$

Utilizând descompunerea lui A după valorile proprii, pentru cazul valorilor proprii distincte, se poate scrie $A = U^{-1}\Lambda U$, cu Λ matricea diagonală a valorilor proprii. Atunci $A^{n-1} = U^{-1}\Lambda^{n-1}U$. Arătati că se poate scrie

$$\mathbf{X}(n) = \lambda_1^{n-1}\,\mathbf{Y}_1 + \lambda_2^{n-1}\,\mathbf{Y}_2 + \lambda_3^{n-1}\,\mathbf{Y}_3$$

în care $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$ nu depind de n. Pentru n mare, această sumă este dominată de termenul cel mai mare. Demonstrati că pentru $i = 1, 2, 3$ are loc

$$\frac{1}{n}\log X_i(n) \to \log\lambda$$

cu λ valoarea proprie (pozitivă) ce mai mare. Astfel, pentru n mare, numărul de secvente de lungime n creste ca λ^n. Calculati λ pentru matricea A de mai sus (cazul în care valorile proprii nu sunt distincte poate fi manipulat în mod similar).

d. Se abordează acum o tratare diferită. Se consideră lantul Markov cu diagrama de stare dată la punctul a. dar cu probabilitătile tranzitiilor arbitrare. Prin urmare matricea de tranzitie a lantului Markov este

$$P = \begin{bmatrix} 0 & \alpha & 1 \\ 1 & 0 & 0 \\ 0 & 1-\alpha & 0 \end{bmatrix}$$

Arătati că distributia stationară a lantului Markov este

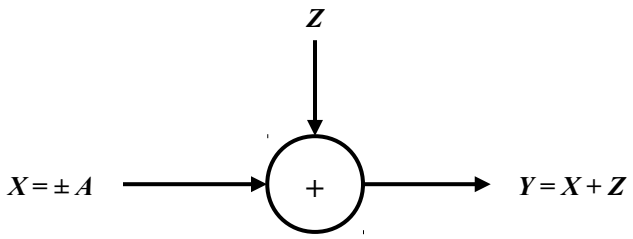
$$\mu = \left[\frac{1}{3-\alpha} \quad \frac{1}{3-\alpha} \quad \frac{1-\alpha}{3-\alpha}\right]^T$$

e. Maximizati rata entropiei lantului Markov în raport cu α. Care este entropia maximă a lantului?

f. Comparati rata maximă a entropiei obtinută la punctul e. cu $\log\lambda$ de la punctul c. De ce cele două răspunsuri sunt identice?

Problema 49. Un canal de comunicatie

Figura alăturată reprezintă schematic un canal de comunicatie.



Intrarea este pe două niveluri, fie $X = A$, fie $X = -A$, cu A o amplitudine de stabilit în cursul solutionării problemei. Canalul este afectat de zgomotul aditiv Z, gaussian cu media 0 si dispersia 100.

a. Găsiti constanta K astfel încât

$$\Pr(Y < -A + K | X = -A) = 0{,}90$$
$$\Pr(A - K \le Y | X = A) = 0{,}90$$

Valoarea K se poate calcula chiar dacă A nu a fost încă evaluat. (*Indicatie*: se utilizează faptul că dacă $X = -A$ atunci distributia conditionată a lui Y este gaussiană cu media $-A$ si cu dispersia 100 si dacă $X = A$ atunci

distributia conditionată a lui Y este gaussiană cu media A si cu dispersia 100).

Solutie: Repartitia valorilor unei variabile aleatoare X normală (gaussiană) este descrisă de functia densitate de probabilitate

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-m)^2}{2\sigma^2}}$$

definită pe întreaga axă reală, depinzând de parametrii m (media) si σ^2 (dispersia). Acesti parametri sunt în cazul de fată, respectiv, $\pm A$ si 100.

Prima valoare K cerută se calculează prin rezolvarea ecuatiei în x

$$\int_{-\infty}^{x} f(x)dx = 0{,}9$$

punând în $f(x)$, $m = -A$.

A doua valoare K cerută se calculează prin rezolvarea unei alte ecuatii în x

$$1 - \int_{-\infty}^{x} f(x)dx = 0{,}9$$

punând în $f(x)$ de data aceasta, $m = A$.

O integrală de genul celor care apar în cele două ecuatii este cunoscută în literatură ca functia de repartitie $F(x)$ egală cu probabilitatea ca variabila aleatoare X să ia valori inferioare valorii x. Valorile functiei de repartitie pentru valori x variate se pot calcula – de pildă cu functia Matlab **`normcdf`** sau, invers, cu functia **`norminv`** –, dar sunt si tabelate în cărti pentru o variabilă aleatoare normală normată (de medie nulă si de dispersie egală cu unitatea). Un apel Matlab

```
x = norminv(0.9, 0, 10)
```

produce valoarea **`x = 12.8155`**, iar un apel

```
x = norminv(0.1, 0, 10)
```

produce valoarea **`x = - 12.8155`**. Acestea sunt valorile pentru care functia de repartitie ia valoarea 0,9, respectiv 0,1, atunci când media variabilei este zero si abaterea medie pătratică este $\sigma = 10$. O deplasare a mediei în $-A$ sau în $+A$ deplasează si valorile x cu aceleasi cantităti. Este usor de constatat că ambele ecuatii au solutia $K = 12{,}8155$.

b. Alegeti nivelul A, acel număr real pozitiv minim pentru care evenimentele $\{Y < -A + K\}$ si $\{A - K \le Y\}$ nu se pot produce simultan.

Solutie: Cele două intervale de valori care corespund celor două evenimente trebuie să fie disjuncte. La limită, este permis ca $-A + K = A - K$, adică $A = K = 12{,}8155$.

c. Pentru numerele K si A stabilite mai devreme, se formează din observatiile Y estimările $\hat{X}$ ale intrărilor X ale canalului conform relatiei

$$\hat{X}: \begin{cases} \cdot A & Y \langle \cdot A \dashv K \\ A & \text{altfel} \end{cases}$$

Intuitiv, estimarea $\hat{X}$ a lui X ar trebui să fie corectă în proportie de 90% din cazuri. Verificati această posibilitate după procedura care urmează:

- Utilizati Matlab-ul pentru a alege pseudoaleator un vector **`x`** de 10.000 de intrări ale canalului simulate, astfel încât fiecare intrare aleasă la întâmplare să fie din multimea cu două elemente $\{-A, A\}$.
- Utilizati Matlab-ul pentru a genera pseudoaleator cu functia Matlab **`randn`** un vector **`z`** de 10.000 de valori ale componentei de zgomot din canal.
- Formati vectorul **`y = x + z`** de 10.000 de iesiri simulate.
- Formati un vector **`xhat`** de 10.000 de estimări ale intrărilor canalului, fiecare formată din componenta **`y`** potrivit aplicatiei $Y \rightarrow \hat{X}$ de mai sus.
- Estimati probabilitatea Pr($X = \hat{X}$) prin evaluarea frecventei relative cu care elementele din **`x`** sunt egale cu elementele corespunzătoare din **`xhat`**.

Solutie: Se propune următorul script Matlab:

```
clear
a=norminv(0.9,0,10);
n=10000;
x=unidrnd(2,1,n);
x=2*a*(x-1.5);
z=normrnd(0,10,1,n);
y=x+z;
nc=0;
for i=1:n
    if x(i)*y(i)>0
        nc=nc+1;
    end
end
nc
nc/n
```

CODAREA PENTRU CANALE FĂRĂ PERTURBATII

Lucrarea 5.

Tema 1. Algoritmul Huffman pentru coduri compacte

Se consideră o sursă fără memorie generatoare a n simboluri, $2 \leq n \leq 15$. Simbolurile pot fi chiar numerele naturale de la 1 la n.
Programul Matlab care urmează este implementarea algoritmului Huffman în cazul binar si se disting cu usurintă câteva etape de calcul.
Se alege mai întâi numărul n de simboluri, printr-o modificare în linia a doua a programului.
Se generează aleator n numere subunitare, probabilitătile de aparitie a celor n simboluri. Probabilitătile se ordonează descrescător, asa cum cere algoritmul Huffman.
Se parcurge partea directă a algoritmului când se realizează succesiv sursele reduse, cu grija de a fi si ordonate. Se ajunge în cele din urmă la o sursă redusă cu numai două simboluri.
Urmează parcursul invers în cadrul căruia se atribuie efectiv, pas cu pas, coduri simbolurilor sursei.
Se afisează cuvintele alocate ale codului alături de probabilitătile simbolurilor asociate.
Se evaluează lungimea medie a codului si entropia sursei. Se poate face astfel o comparatie între lungimea medie a codului cu entropia sursei.
Se recomandă repetarea executării calculelor pentru surse diferite între ele prin lista de probabilităti si/sau prin numărul n de simboluri generate.
Orice ameliorare a programului propus este apreciată.

```
clear
n=15;    % numarul de simboluri ale sursei
if n>15
    n=15;
end    % o limitare a numarului de simboluri
if n<2
    n=2;
end    % o alta limitare a numarului de simboluri
simb=1:n;    % simbolurile sursei
prob=rand(1,n);    % generarea a n numere aleatoare intre 0 si 1
prob=prob/sum(prob);   % normaliarea listei de n numere; suma=1
%prob=(1/n)*ones(1,n);
% prin decomentare, probabilitatile se fac egale
i=zeros(n,n);
lun=i;
num=i;
y=i;
[y(1,:),i1]=sort(prob,'descend');    % ordonarea probabiltatilor
for j=1:n
    simb1(j)=simb(i1(j));
end
simboluri=simb
probabilitati=prob
simboluri_ordonate=simb1
```

```
probabilitati_ordonate=y(1,:)
i(1,:)=1:n;
for j=2:(n-1)
    y(j,:)=y(j-1,:);
    y(j,n-j+1)=y(j-1,n-j+2)+y(j-1,n-j+1);
    y(j,n-j+2)=0;
    [y(j,1:(n-j+1)),i(j,1:(n-j+1))]=sort(y(j,1:(n-
j+1)),'descend');
end    % secventa directa a algoritmului Huffman
lun(n-1,1)=1;lun(n-1,2)=1;
num(n-1,1)=0;num(n-1,2)=1;
for j=(n-1):(-1):2
    [l,m]=max(i(j,:));
    %[l m]
    i2=0;
    for k=1:(n-j+1)
        if k~=m
            i2=i2+1;
            lun(j-1,i2)=lun(j,k);
            num(j-1,i2)=num(j,k);
        end
    end
    lun(j-1,l)=lun(j,m)+1;
    lun(j-1,l+1)=lun(j,m)+1;
    num(j-1,l)=2*num(j,m);
    num(j-1,l+1)=2*num(j,m)+1;
end    % secventa inversa a algoritmului Huffman
lungimi_cuvinte_de_cod=lun(1,:)
% afisarea lungimilor cuvintelor de cod
cuvinte_de_cod='';
for j=1:n
    cuvinte_de_cod=[cuvinte_de_cod,'
',dec2bin(num(1,j),lun(1,j))];
end
cuvinte_de_cod    % afisarea cuvintelor de cod Huffman
entropia_sursei=-y(1,:)*log((y(1,:))')/log(2)
% entropia sursei
lungimea_medie_a_codului=lun(1,:)*(y(1,:))'
% lungimea medie a codului
```

Tema 2. Un cod Huffman în acțiune

Se selectează un fisier de un tip anumit. Numele lui (si calea dacă este cazul) sunt înlocuite în linia a doua a programului. În varianta alăturată, este trecut sirul `'TCI5.doc'` care presupune că fisierul este în acelasi folder/director cu programul.
Se citeste fisierul ales ca o succesiune de octeti (bytes). Se face statistica fisierului considerat sursă de informatie, sursă de simboluri ASCII

corespunzătoare unor numere de la 0 la 255. Frecventele relative sunt considerate probabilități. Fisierul este considerat "fără memorie'.
Se aplică algoritmul Huffman care duce la alocarea unor cuvinte de cod fiecărui cod ASCII.
Se calculează lungimea medie a codului, se calculează eficienta codului.
Se recomandă executarea programului pentru mai multe fisiere de acelasi tip (aceeasi extensie), pentru fisiere de tipuri variate. Se compară eficienta codurilor în cazurile studiate, se compară si cu eficienta comprimării fisierelor cu programele de comprimare comerciale.

```
clear
idf=fopen('TCI5.doc');
% se creaza un identificator pentru fisierul ales
[a,n]=fread(idf);
% se citeste fisierul ca o secventa de n bytes
st=fclose(idf);     % se inchide fisierul
p=zeros(256,256);
f=zeros(1,256);iper=f;
for i=1:n           % se calculeaza frecventele absolute
   f(a(i)+1)=f(a(i)+1)+1;
end
sf=sum(f);
f=f/sf;             % se calculeaza frecventele relative
h=0;                % se calculeaza entropia
for i=1:256
   if f(i)>0
      h=h-f(i)*log(f(i));
   end
end
h=h/log(2);
% incepe comprimarea Huffman
[y,in]=sort(-f);
p(:,1)=-y';
for k=2:255
% se realizeaza "redusele" (parcursul direct)
    z=p(258-k,k-1)+p(257-k,k-1);
    p(257-k,k)=z;
    p(258-k,k)=0;
    p(1:256-k,k)=p(1:256-k,k-1);
    [y,inc]=sort(-p(:,k));
    p(:,k)=-y;
    ina=(1:256)';
    iper(k)=1;
    for i=1:257-k
% se localizeaza simbolul rezultat din alte doua
        if ina(i)==inc(i)
            iper(k)=i+1;
        end
        iper(k)=min(iper(k),257-k);
    end
end
% parcursul invers
```

```
lcod=zeros(256,1);
lcod(1)=1;lcod(2)=1;nc=1;
for k=254:(-1):1
    nc=nc+1;
    lcper=lcod(iper(k+1));
    for i=1:nc
        if i>iper(k+1)
            lcod(i-1)=lcod(i);
        end
    end
    lcod(nc)=lcper+1;
    lcod(nc+1)=lcod(nc);
end
lmed=lcod'*p(:,1);
'Entropia si lungimea medie a codului (biti):',h,lmed
'eficienta codului (%):',efi=100*h/lmed
```

Lucrarea 6. Codarea aritmetică

Suportul teoretic al codării aritmetice si multe aspecte de ordin practic ale acestui gen de codare sunt foarte bine explicate în lucrarea *Arithmetic Coding revealed. A guided tour from theory to praxis* de E.Bodden, M.Clasen si J.Kneis. Lucrarea este bogat citată în *Notele de curs* la disciplina *Transmiterea si codarea informatiei*.
În Anexele lucrării lui E.Bodden, M.Clasen si J.Kneis este dată si o implementare a codării aritmetice scrisă în C++. Lectura lucrării citate si parcurgerea Anexelor ei reprezintă o invitatie la a experimenta si a aprofunda tema atât de generoasă a codării aritmetice.
Din motive de simplitate, în lucrarea prezentă se sugerează utilizarea a două implemetări în Matlab, una realizată de autor, cealaltă adaptată din literatură.

Tema 1. Codarea aritmetică în numere reale

Prima temă a acestei lucrări cuprinde codarea aritmetică prin numere reale si, numai ca tentativă, o transformare a codului real în cod binar.
Programul didactic **`arit1`** execută codarea aritmetică a unor mesaje scurte, de până la 10 caractere, prin numere reale. Sursa de informatie este dată printr-un model static, asadar probabilitătile asociate caracterelor sunt mereu aceleasi. Aceste probabilităti sunt generate aleator în primele linii ale programului. Probabilitătile se pot initializa si după dorintă, introducând o linie de program specială după initilaizarea aleatoare.

```
% Program ARIT1 de codare aritmetica prin numere reale
% a mesajelor scurte
clear
n=5;  % Numarul de caractere din alfabetul sursei (la alegere)
p=rand(1,n);
p=p/sum(p);  % Generarea de probabilitati pentru sursa
disp('Probabiltatile celor n simboluri si entropia sursei')
```

```
[p entropie(p)]
cum(1)=0;
for i=1:n
    cum(i+1)=sum(p(1:i));
end  % Calculul vectorului probabilitatilor cumulate
disp('Vectorul probabilitatilor cumulate')
cum
low=cum(1);high=cum(n+1);  % Initializarea limitelor
intervalului
m=10;  % Lungimea mesajului (la alegere)
disp('Un mesaj de m simboluri')
mesaj=unidrnd(n,1,m)  % Generarea aleatoare a mesajului
for k=1:m
    i=mesaj(k);
    lowprim=low+(high-low)*cum(i);
    highprim=low+(high-low)*cum(i+1);
    nc=num2str(k,3);
    disp(strcat(nc,' caractere codate; limitele intervalului
dupa codare'))
    format long, [lowprim highprim]
    low=lowprim;
    high=highprim;
end  % Calculul intervalului care contine codul real
codul_real=(low+high)/2
disp('Lungimea codului')
nb=floor(log2(1./(high-low)))+1
disp('Lungimea medie a codului')
nb/m
disp('Expresia binara a limitelor intervalului si a codului pe
40 de biti')
bs=real2bin(high,40);
bm=real2bin((low+high)/2,40);
bi=real2bin(low,40);
[bs; bm; bi]
disp('Codul se limiteaza la nb biti')
```

Programul **arit2** elaborat tot în scop didactic parcurge algoritmul de decodare a unui cod aritmetic. El preia de la programul **arit1** executat în prealabil, lista cumulativă a probabilităților asociate simbolurilor sursei, codul ca număr real, lungimea mesajului codat precum si codul binar obtinut prin conversia pe un număr limitat de biti (cel mult 40) a codului real.
Conform comentariilor inserate în program, rezultă două secvente decodate, reconstituite din cele două coduri utilizate, unul ca număr real, altul ca număr real reconstituit dintr-o secventă finită de biti.

```
% Program ARIT2 de decodare a unui cod aritmetic
% Programul preia de la programul ARIT1 lungimea mesajului m,
% vectorul cum, valoarea codul_real si codul binar.
cum1=cum
codul_real
lungimea_mesajului=m
```

```
bm1=bm(1:(nb+2));  % Codul binar al mesajului limitat la nb+2 <=
40 biti
v=codul_real;
[i,n]=size(cum);
n=n-1;
high=1;low=0;  % Se initilaizeaza limitele intervalului
for k=1:m
    cum1=low*ones(1,n+1)+(high-low)*cum;
    for i=1:n
        if v>=cum1(i) & v<cum1(i+1)
            mes(k)=i;
            low=cum1(i);
            high=cum1(i+1);
        end
    end
end  % Bucla de decodare
mesajul_decodat=mes
codul_real_din_binar=bin2real(bm1)
v=codul_real_din_binar;
[i,n]=size(cum);
n=n-1;
high=1;low=0;  % Se initilaizeaza limitele intervalului
for k=1:m
    cum1=low*ones(1,n+1)+(high-low)*cum;
    for i=1:n
        if v>=cum1(i) & v<cum1(i+1)
            mes(k)=i;
            low=cum1(i);
            high=cum1(i+1);
        end
    end
end  % Bucla de decodare din codul binar
mesajul_decodat=mes
```

Programele (script-urile) **arit1** si **arit2** folosesc si următoarele trei functii Matlab scrise de autor. Două din acestea fac oprațiile de conversie mentionate în subtitluri. A treia evaluează entropia pentru o sursă dată prin vectorul de probabilităti specific si a mai fost utilizată si la alte lucrări anterioare.

```
function alfa=bin2real(b)
% functie de conversie binar (digitii de dupa punctul binar)
% la real
[m,n]=size(b);
a=0;
x=1/2;
for i=1:n
    a=x*a+b(n-i+1);
end
alfa=a;

function b=real2bin(a,n)
% Functie de conversie de la real pozitiv subunitar
% la binar (n digiti dupa punctul binar)
```

```
prov=zeros(1,n);
a=a/2;
for i=1:n
    prov(i)=floor(2*a);
    a=2*a-prov(i);
end
b=prov;

function entropie=ent(s)
% s este vectorul (linie) al probabilitatilor
[n,m]=size(s);
ent=0;
for i=1:m
    if s(i)>0
        ent=ent-s(i)*log(s(i));
    end
end
ent=ent/log(2);
entropie=ent;
```

Tema 2. Codarea aritmetică în numere întregi

Următoarele două functii Matlab, **arithenco** si **arithdeco** sunt preluate din capitolul Matlab-ului 7.1 denumit **toolbox\comm\comm**. Sursa este descrisă aici prin frecvențele absolute (numere întregi) ale caracterelor generate.
O lectură atentă a comentariilor din programe face posibilă utilizarea corectă a acestor functii de codare si de decodare pentru codurile aritmetice.
Deoarece functia **arithdeco** preia rezultatele codării efectuate de functia **arithenco**, se propune imediat un script **ari** al autorului care cuplează operatiile de codare si decodare realizate cu cele două functii Matlab de bibliotecă.
Se sugerează o comparatie a rezultatelor codării/decodării cu aceste functii, cu rezultatele obtinute cu programele de la **Tema 1** elaborate de autor.

```
% Program ilustrativ ARI de utilizare a functiilor Matlab
% arithenco si arithdeco pentru codarea aritmetica în
% numere intregi si decodarea unui cod aritmetic
clear
ls=7;  % Lungimea secventei codate (la alegere)
cs=5;  % Numarul simbolurilor sursei (la alegere)
seq=unidrnd(cs,1,ls);  % Generarea secventei de codat
sir_de_codat=seq
counts=unidrnd(10*cs,1,cs)  % Genrarea vectorului frecventelor
code = arithenco(seq, counts)  % Codarea
[i,lungime]=size(code);
lungimi=[lungime lungime/ls]
ent=entropie(counts/sum(counts))
sir_recuperat = arithdeco(code,counts,ls)  % Decodarea

function code = arithenco(seq, counts)
```

```
%ARITHENCO Encode a sequence of symbols using arithmetic coding.
%   CODE = ARITHENCO(SEQ, COUNTS) generates binary arithmetic
code
%   corresponding to the sequence of symbols specified in the
vector SEQ.
%   The vector COUNTS contains the symbol counts (the number of
times each
%   symbol of the source's alphabet occurs in a test data set)
and represents
%   the source's statistics.
%
%   Example:
%     Consider a source whose alphabet is {x, y, z}. A 177-
symbol test data
%     set from the source contains 29 x's, 48 y's and 100 z's.
To encode the
%     sequence yzxzz, use these commands:
%
%       seq = [2 3 1 3 3];
%       counts = [29 48 100];
%       code = arithenco(seq, counts)
%
%   See also ARITHDECO.

%   Copyright 1996-2002 The MathWorks, Inc.
%   $Revision: 1.3 $ $Date: 2002/06/17 12:22:10 $

%   References:
%         [1] Sayood, K., Introduction to Data Compression,
%         Morgan Kaufmann, 2000, Chapter 4, Section 4.4.3.

% Input argument checks
error(nargchk(2, 2, nargin));

% Output argument checks
error(nargoutchk(0, 1, nargout));

% Check parameters
eStr = errorchk(seq, counts);
if (eStr.ecode == 1)
    error(eStr.emsg);
end

% Check the incoming orientation and adjust if necessary
[row_s, col_s] = size(seq);
if (row_s > 1),
    seq = seq.';
end

[row_c, col_c] = size(counts);
if (row_c > 1),
    counts = counts.';
end
```

```
% Compute the cumulative counts vector from the counts
cum_counts = [0, cumsum(counts)];

% Compute the Word Length required.
total_count = cum_counts(end);
N = ceil(log2(total_count)) + 2;

% Initialize the lower and upper bounds.
dec_low = 0;
dec_up = 2^N-1;
E3_count = 0;

% Obtain an over estimate for the length of CODE and initialize
CODE
code_len = length(seq) * ( ceil(log2(length(counts))) + 2 ) + N;
code = zeros(1, code_len);
code_index = 1;

% Loop for each symbol in SEQ
for k = 1:length(seq)

    symbol = seq(k);
    % Compute the new lower bound
    dec_low_new = dec_low + floor( (dec_up-
dec_low+1)*cum_counts(symbol+1-1)/total_count );

    % Compute the new upper bound
    dec_up = dec_low + floor( (dec_up-
dec_low+1)*cum_counts(symbol+1)/total_count )-1;

    % Update the lower bound
    dec_low = dec_low_new;

    % Check for E1, E2 or E3 conditions and keep looping as long
as they occur.
    while( isequal(bitget(dec_low, N), bitget(dec_up, N)) || ...
        (isequal(bitget(dec_low, N-1), 1) &&
isequal(bitget(dec_up, N-1), 0) ) ),

        % If it is an E1 or E2 condition,
        if isequal(bitget(dec_low, N), bitget(dec_up, N)),

            % Get the MSB
            b = bitget(dec_low, N);
            code(code_index) = b;
            code_index = code_index + 1;

            % Left shifts
            dec_low = bitshift(dec_low, 1) + 0;
            dec_up = bitshift(dec_up, 1) + 1;

            % Check if E3_count is non-zero and transmit
appropriate bits
            if (E3_count > 0),
```

```
                % Have to transmit complement of b, E3_count
times.
                code(code_index:code_index+E3_count-1) =
bitcmp(b, 1).*ones(1, E3_count);
                code_index = code_index + E3_count;
                E3_count = 0;
            end

            % Reduce to N for next loop
            dec_low = bitset(dec_low, N+1, 0);
            dec_up  = bitset(dec_up, N+1, 0);

        % Else if it is an E3 condition
        elseif ( (isequal(bitget(dec_low, N-1), 1) && ...
            isequal(bitget(dec_up, N-1), 0) ) ),

            % Left shifts
            dec_low = bitshift(dec_low, 1) + 0;
            dec_up  = bitshift(dec_up, 1) + 1;

            % Reduce to N for next loop
            dec_low = bitset(dec_low, N+1, 0);
            dec_up  = bitset(dec_up, N+1, 0);

            % Complement the new MSB of dec_low and dec_up
            dec_low = bitxor(dec_low, 2^(N-1) );
            dec_up  = bitxor(dec_up, 2^(N-1) );

            % Increment E3_count to keep track of number of
times E3 condition is hit.
            E3_count = E3_count+1;
        end
    end
end

% Terminate encoding
bin_low = de2bi(dec_low, N, 'left-msb');
if E3_count==0,
    % Just transmit the final value of the lower bound bin_low
    code(code_index:code_index + N - 1) = bin_low;
    code_index = code_index + N;
else
   % Transmit the MSB of bin_low.
   b = bin_low(1);
   code(code_index) = b;
   code_index = code_index + 1;

   % Then transmit complement of b (MSB of bin_low), E3_count
times.
   code(code_index:code_index+E3_count-1) = bitcmp(b,
1).*ones(1, E3_count);
   code_index = code_index + E3_count;

   % Then transmit the remaining bits of bin_low
```

```
   code(code_index:code_index+N-2) = bin_low(2:N);
   code_index = code_index + N - 1;
end

% Output only the filled values
code = code(1:code_index-1);

% Set the same output orientation as seq
if (row_s > 1)
    code = code.';
end

%-------------------------------------------------------------
function eStr = errorchk(seq, counts)
% Function for validating the input parameters.

eStr.ecode = 0;
eStr.emsg = '';

% Check to make sure a vector has been entered as input and not
a matrix
if (length(find(size(seq)==1)) ~= 1)
    eStr.emsg = ['The symbol sequence parameter must be a vector
of positive ',...
                 'finite integers.'];
    eStr.ecode = 1; return;
end

% Check to make sure a character array is not specified for SEQ
if ischar(seq)
    eStr.emsg = ['The symbol sequence parameter must be a vector
of positive ',...
                 'finite integers.'];
    eStr.ecode = 1; return;
end

% Check to make sure that finite positive integer values (non-
complex) are
% entered for SEQ
if ~all(seq > 0) || ~all(isfinite(seq)) || ~isequal(seq,
round(seq)) || ...
    ~isreal(seq)
    eStr.emsg = ['The symbol sequence parameter must be a vector
of positive ',...
                 'finite integers.'];
    eStr.ecode = 1; return;
end

if length(find(size(counts)==1)) ~= 1
    eStr.emsg = ['The symbol counts parameter must be a vector
of positive ',...
                 'finite integers.'];
    eStr.ecode = 1; return;
end
```

```
% Check to make sure that finite positive integer values (non-
complex) are
% entered for COUNTS
if ~all(counts > 0) || ~all(isfinite(counts)) ||
~isequal(counts, round(counts)) || ...
     ~isreal(counts)
    eStr.emsg = ['The symbol counts parameter must be a vector
of positive ',...
                 'finite integers.'];
    eStr.ecode = 1; return;
end

% Check to ensure that the maximum value in the SEQ vector is
less than or equal
% to the length of the counts vector COUNTS.
if max(seq) > length(counts)
    eStr.emsg = ['The symbol sequence parameter can take values
only between',...
                 ' 1 and the length of the symbol counts
parameter.'];
    eStr.ecode = 1; return;
end

% [EOF]

function dseq = arithdeco(code, counts, len)
%ARITHDECO Decode binary code using arithmetic decoding.
%   DSEQ = ARITHDECO(CODE, COUNTS, LEN) decodes the binary
arithmetic code
%   in the vector CODE (generated using ARITHENCO) to the
corresponding
%   sequence of symbols. The vector COUNTS contains the symbol
counts (the
%   number of times each symbol of the source's alphabet occurs
in a test
%   data set) and represents the source's statistics. LEN is the
number of
%   symbols to be decoded.
%
%   Example:
%     Consider a source whose alphabet is {x, y, z}. A 177-
symbol test data
%     set from the source contains 29 x's, 48 y's and 100 z's.
To encode the
%     sequence yzxzz, use these commands:
%
%       seq = [2 3 1 3 3];
%       counts = [29 48 100];
%       code = arithenco(seq, counts)
%
%     To decode this code (and recover the sequence of
%     symbols it represents) use this command:
%
```

```
%       dseq = arithdeco(code, counts, 5)
%
%   See also ARITHENCO.

%   Copyright 1996-2002 The MathWorks, Inc.
%   $Revision: 1.2 $ $Date: 2002/04/14 20:12:32 $

%   References:
%         [1] Sayood, K., Introduction to Data Compression,
%         Morgan Kaufmann, 2000, Chapter 4, Section 4.4.3.

% Input argument check
error(nargchk(3, 3, nargin));

% Output argument check
error(nargoutchk(0, 1, nargout));

% Check parameters
eStr = errorchk(code, counts, len);
if (eStr.ecode == 1)
    error(eStr.emsg);
end

% Check the incoming orientation and adjust if necessary
[row_cd, col_cd] = size(code);
if (row_cd > 1),
    code = code.';
end

[row_c, col_c] = size(counts);
if (row_c > 1),
    counts = counts.';
end

% Compute the cumulative counts vector from the counts vector
cum_counts = [0, cumsum(counts)];

% Compute the Word Length (N) required.
total_count = cum_counts(end);
N = ceil(log2(total_count)) + 2;

% Initialize the lower and upper bounds.
dec_low = 0;
dec_up = 2^N-1;

% Read the first N number of bits into a temporary tag bin_tag
bin_tag = code(1:N);
dec_tag = bi2de(bin_tag, 'left-msb');

% Initialize DSEQ
dseq = zeros(1,len);
dseq_index = 1;

k=N;
```

```
ptr = 0;

% This loop runs untill all the symbols are decoded into DSEQ
while (dseq_index <= len)

    % Compute dec_tag_new
    dec_tag_new =floor( ((dec_tag-dec_low+1)*total_count-1)/
(dec_up-dec_low+1) );

    % Decode a symbol based on dec_tag_new
    ptr = pick(cum_counts, dec_tag_new);

    % Update DSEQ by adding the decoded symbol
    dseq(dseq_index) = ptr;
    dseq_index = dseq_index + 1;

    % Compute the new lower bound
    dec_low_new = dec_low + floor( (dec_up-
dec_low+1)*cum_counts(ptr-1+1)/total_count );

    % Compute the new upper bound
    dec_up = dec_low + floor( (dec_up-
dec_low+1)*cum_counts(ptr+1)/total_count )-1;

    % Update the lower bound
    dec_low = dec_low_new;

    % Check for E1, E2 or E3 conditions and keep looping as long
as they occur.
     while ( isequal(bitget(dec_low, N), bitget(dec_up, N))
| ...
        ( isequal(bitget(dec_low, N-1), 1) &
isequal(bitget(dec_up, N-1), 0) ) ),

        % Break out if we have finished working with all the
bits in CODE
        if ( k==length(code) ), break, end;
        k = k + 1;

        % If it is an E1 or E2 condition, do
        if isequal(bitget(dec_low, N), bitget(dec_up, N)),

            % Left shifts and update
            dec_low = bitshift(dec_low, 1) + 0;
            dec_up  = bitshift(dec_up,  1) + 1;

            % Left shift and read in code
            dec_tag = bitshift(dec_tag, 1) + code(k);

            % Reduce to N for next loop
            dec_low = bitset(dec_low, N+1, 0);
            dec_up  = bitset(dec_up,  N+1, 0);
            dec_tag = bitset(dec_tag, N+1, 0);
```

```
        % Else if it is an E3 condition
        elseif ( isequal(bitget(dec_low, N-1), 1) & ...
            isequal(bitget(dec_up, N-1), 0) ),

            % Left shifts and update
            dec_low = bitshift(dec_low, 1) + 0;
            dec_up  = bitshift(dec_up,  1) + 1;

            % Left shift and read in code
            dec_tag = bitshift(dec_tag, 1) + code(k);

            % Reduce to N for next loop
            dec_low = bitset(dec_low, N+1, 0);
            dec_up  = bitset(dec_up,  N+1, 0);
            dec_tag = bitset(dec_tag, N+1, 0);

            % Complement the new MSB of dec_low, dec_up and
dec_tag
            dec_low = bitxor(dec_low, 2^(N-1) );
            dec_up  = bitxor(dec_up,  2^(N-1) );
            dec_tag = bitxor(dec_tag, 2^(N-1) );

        end
    end % end while
end % end while length(dseq)

% Set the same output orientation as code
if (row_cd > 1)
    dseq = dseq.';
end
%---------------------------------------------------------------
function [ptr] = pick(cum_counts, value);
% This internal function is used to find where value is
positioned

% Check for this case and quickly exit
if value == cum_counts(end)
    ptr = length(cum_counts)-1;
    return
end

c = find(cum_counts <= value);
ptr = c(end);

%---------------------------------------------------------------
function eStr = errorchk(code, counts, len);
% Function for validating the input parameters.

eStr.ecode = 0;
eStr.emsg = '';

% Check to make sure a vector has been entered as input and not
a matrix
if (length(find(size(code)==1)) ~= 1)
```

```
    eStr.emsg = ['The binary arithmetic code parameter must be a
double ',...
                 'array of zeros and ones only.'];
    eStr.ecode = 1; return;
end

if (length(find(size(counts)==1)) ~= 1)
    eStr.emsg = ['The symbol counts parameter must be a vector
of positive ',...
                 'finite integers.'];
    eStr.ecode = 1; return;
end

% Check to make sure that CODE is binary
if any(code ~= 1 & code ~= 0)
    eStr.emsg = ['The binary arithmetic code parameter must be a
double ',...
                 'array of zeros and ones only.'];
    eStr.ecode = 1; return;
end

% Check to make sure that finite positive integer values (non-
complex) are
% entered for COUNTS
if ~all(counts > 0) | ~all(isfinite(counts)) | ~isequal(counts,
round(counts)) | ...
    ~isreal(counts)
    eStr.emsg = ['The symbol counts parameter must be a vector
of positive ',...
                 'finite integers.'];
    eStr.ecode = 1; return;
end

% Check to make sure LEN is scalar
if ~isequal(size(len),[1 1])
    eStr.emsg = 'The number of symbols must be a positive,
finite, integer scalar.';
    eStr.ecode = 1; return;
end

% Check to make sure that finite positive integer value (non-
complex) is
% entered for LEN
if len < 1 | ~isfinite(len) | (len ~= round(len)) | ~isreal(len)
    eStr.emsg = 'The number of symbols must be a positive,
finite, integer scalar.';
    eStr.ecode = 1; return;
end

% EOF
```

Problema 50. Coduri nepotrivite

Care din codurile următoare nu pot fi coduri Huffman oricare ar fi setul de probabilităti ale simbolurilor sursei?

a. {1, 01, 00}

Solutie: Codurile Huffman sunt coduri compacte. Ele satisfac dubla inegalitate a lui Shannon

$$H(X) \le L < H(X) + 1$$

sau

$$0 \le L - H(X) < 1$$

cu L lungimea medie a codului si cu $H(X)$ entropia sursei codate.

Se admite că probabilitătile asociate simbolurilor sursei sunt $[p, (1-p)/2 + \alpha, (1-p)/2 - \alpha]$, cu α un număr care să nu facă probabilitătile numere negative sau supraunitare. Se evaluează lungimea medie

$$L = 1\cdot p + 2\cdot[(1-p)/2 + \alpha] + 2\cdot[(1-p)/2 - \alpha] = 2 - p$$

si entropia

$$H(X) =$$

$$= -p\log p - [(1-p)/2 + \alpha]\log[(1-p)/2 + \alpha] - [(1-p)/2 - \alpha]\log[(1-p)/2 - \alpha]$$

Suma ultimilor doi termeni este maximă atunci când $\alpha = 0$, adică atunci când cele două simboluri sunt egal probabile, astfel că pentru un p dat, entropia maximă este

$$H_{\max}(X) = -p\log p - 2[(1-p)/2]\log[(1-p)/2] =$$
$$= -p\log p - (1-p)\log(1-p) + 1 - p = 1 - p + H(p)$$

Prima parte a dublei inegalităti Shannon modificate devine în cazul critic $H_{\max}(X)$

$$L - H_{\max}(X) = 1 - H(p) \ge 0$$

cu $H(p)$ entropia unei surse binare cu probabilitatea unuia dintre simboluri p. Deoarece $0 \le H(p) \le 1$, inegalitatea este satisfăcută pentru orice p.

Pentru partea a doua a inegalitătii lui Shannon trebuie identificată, tot asa, situatia cea mai defavorabilă. Aceasta se produce pentru $\alpha = (1-p)/2$. Entropia este în acest caz

$$H_{\min}(X) = -p\log p - (1-p)\log(1-p) = H(p)$$

Se testează acum inegalitatea

$$L - H(X) \le L - H_{\min}(X) = 2 - p - H(p) < 1$$

La extreme, $p = 0$, $p = 1$, expresia în p ia valorile 2, respectiv 1. Nici una nu satisface a doua parte a inegalitătii lui Shannon. Acestea sunt contraexemple la afirmatia "codul propus este Huffman pentru orice listă de probabilităti". Asadar, codul nu poate fi de tipul Huffman pentru orice set de probabilităti.

b. {00, 01, 10, 110}

Solutie: La fel ca punctul anterior poate fi tratat si acest punct al problemei. Există însă o cale mai directă de tratare. Se observă că proprietatea de lungime medie minimă a codului propus nu este îndeplinită. De pildă, pentru orice probabilitate a ultimului dintre simboluri nenulă, codul {00, 01, 10, 11} este mai scurt în medie fată de codul dat. Codul nu este compact si de aceea nu poate fi un cod Huffman oricare ar fi probabilitătile simbolurilor

c. {01, 10}

Solutie: Aici lungimea medie este $L_c = 2$. Dubla inegalitate a lui Shannon, $1 \le L_c < 2$ nu este satisfăcută. Există totdeauna un cod de lungime medie mai mică decât 2, de pildă codul $\{0, 1\}$. Codul propus nu este un cod Huffman pentru toate probabilitătile posibile ale simbolurilor deoarece nu este compact.

Problema 51. Codarea Huffman

Se consideră variabila aleatoare

$$X = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ 0{,}50 & 0{,}26 & 0{,}11 & 0{,}04 & 0{,}04 & 0{,}03 & 0{,}02 \end{pmatrix}$$

a. Găsiti un cod binar Huffman pentru X.
b. Găsiti lungimea medie a cuvântului de cod pentru codul stabilit. Comparati cu entropia.
c. Găsiti un cod ternar Huffman pentru aceeasi variabilă aleatoare X.

Indicatie: Punctele a. si b. se rezolvă aplicând algoritmul Huffman pentru coduri binare, binecunoscut. Punctul c. se rezolvă adaptând algoritmul la cazul ternar: simbolurile din coada topului probabilistic la fiecare etapă se grupează câte trei, sursa redusă finală trebuie să aibă trei simboluri în caz contrar se adaugă la sursa initială simboluri (fictive) de probabilitate nulă în număr strict necesar (unul sau două).

Problema 52. Coduri

Fie variabilele $X_1, X_2, \ldots$, i.i.d.[1] cu

$$X = \begin{cases} 1 & \text{cu probabilitatea } 1/2 \\ 2 & \text{cu probabilitatea } 1/4 \\ 3 & \text{cu probabilitatea } 1/4 \end{cases}$$

Se consideră atribuirea următorului cod

$$C(x) = \begin{cases} 0 & \text{dacă } x = 1 \\ 01 & \text{dacă } x = 2 \\ 11 & \text{dacă } x = 3 \end{cases}$$

a. Este acest cod nesingular?
Solutie: Da.
b. Este unic decodabil?
Solutie: Da.
c. Este instantaneu?
Solutie: Nu. Conditia de prefix nu este satisfăcută.
d. Care este rata entropiei procesului

$$Z_1Z_2Z_3\ldots = C(X_1)C(X_2)C(X_3)\ldots \ ?$$

(Exemplu: $x^n = 2311\ldots$ produce procesul codat $z^n = 011100\ldots$.)
Solutie: În medie, pentru o succesiune de n simboluri generate de sursa X, numărul de zerouri în secventa codată z^n este $(1/2)n + (1/4)n = (3/4)n$, iar

[1] i.i.d. – variabile aleatoare independente si identic distribuite.

numărul de unități este (1/4)n + 2(1/4)n = (3/4)n. Asadar, simbolurile binare sunt echiprobabile.
Rata entropiei pentru sursa secundară cu alfabet binar este de 1 bit/simbol.
Calculul ratei raportată la cele n simbouri codate trece prin evaluarea lungimii medii a codului care este 1(1/2) + 2(1/4) + 2(1/4) = 3/2 biti/simbol.
Entropia sursei X este – (1/2)log(1/2) – (1/4)log(1/4) – (1/4)log(1/4) = 3/2 biti/simbol.
Este un caz de codare perfectă pentru că sursa este specială: probabilitătile simbolurilor sunt puteri întregi ale lui 2.

Problema 53. Coduri de cost minim

Cuvinte ca Run!, Help! sau Fire! sunt scurte nu pentru că sunt utilizate frecvent, ci, probabil, pentru că timpul este pretios în situatiile în care aceste cuvinte sunt necesare.
Se presupune că $X = i$ cu probabilitatea p_i, i = 1, 2, …, m. Fie l_i numărul de simboluri binare în cuvântul de cod asociat cu $X = i$ si fie c_i costul pe literă de cod în aceeasi situatie. Astfel, costul mediu C al descrierii lui X prin cod este $C = \sum_{i=1}^{m} p_i c_i l_i$.

a. Minimizati costul mediu C pe toate lungimile l_1, l_2, …, l_m astfel încât $\sum 2^{-l_i} \le 1$. Ignorati restrictia referitoare la lungimile l_i care ar trebui să fie numere întregi. Evidentiati valorile l_1^*, l_2^*, …, l_m^* care fac costul $C = C^*$ minim.

Solutie: Se "confectionează" o sursă cu un alfabet care are acelasi număr de caractere, dar lista de probabilităti asociate este

$$q_i = \frac{p_i c_i}{\sum_{i=1}^{m} p_i c_i}, \quad i = 1, 2, \ldots, m.$$

Inegalitatea din enunt este inegalitatea lui Kraft. Dacă lungimile cuvintelor de cod ar putea fi $l_i^* = -\log(q_i)$, inegalitatea ar fi verificată chiar cu egalitate. Costul C este minim odată cu "costul"

$$C' = \frac{C}{\sum_{i=1}^{m} p_i c_i}$$

si atinge valoarea entropiei sursei "confectionate" convenabil, $E(q_1, q_2, \ldots, q_m) = E(q)$, multiplicată cu un factor constant specific

$$C^* = \left(\sum_{i=1}^{m} p_i c_i \right) E(q) .$$

b. Cum ati folosi procedura de obtinere a unui cod Huffman pentru a minimiza costul mediu C peste toate codurile unic decodabile? Fie C_{Huffman} acest minim.

Solutie: Se aplică algoritmul Huffman pe lista de probabilităti q_i, $i = 1, 2, \ldots, m$.

c. Arătati că

$$C^* \le C_{\text{Huffman}} \le C^* + \sum_{i=1}^{m} p_i c_i$$

Solutie: Prin împărtirea dublei inegalităti din enunt cu $\sum_{i=1}^{m} p_i c_i$ se obtine dubla inegalitate a lui Shannon pentru sursa "confectionată". Inegalitatea primă se explică prin valorile obligatoriu întregi ale lungimilor l_i, diferite si uzual mai mari fată de valorile reale l_i^* de la punctul a. al problemei.

Problema 54. Entropia relativă este costul codării inadecvate

Fie variabila aleatoare X care ia cinci valori posibile (1, 2, 3, 4, 5}. Se consideră două distributii ale acestei variabile (v.tabelul).

a. Calculati $H(p)$, $H(q)$, $D(p||q)$ si $D(q||p)$.
b. Ultimele două coloane ale tabelului contin coduri pentru variabila X. Verificati că lungimea medie a codului C_1 sub distributia p este egală cu entropia $H(p)$. Asadar, C_1 este optimal pentru p. Verificati că C_2 este un cod optimal pentru q.

Simbol	$p(x)$	$q(x)$	$C_1(x)$	$C_2(x)$
1	1/2	1/2	0	1
2	1/4	1/8	10	100
3	1/8	1/8	110	101
4	1/16	1/8	1110	110
5	1/16	1/8	1111	111

c. Se presupune acum că se utilizează codul C_2 când distributia este p. Care este lungimea medie a cuvântului de cod? Cu cât este mai mare aceasta fată de entropia $H(p)$?
d. Care este pierderea dacă este folosit codul C_1 când distributia este q?

Problema 55. Inegalitatea lui Kraft

a. Găsiti un cod binar ($D = 2$) liber de prefix pentru sursa cu probabilitătile (1/3, 1/5, 1/5, 2/15, 2/15).
b. Găsiti codul liber de prefix optimal, adică acel cod cu lungimea medie minimă. (*Indicatie*: Optimizati lungimea medie cu conditia la frontieră a (in)egalitătii lui Kraft pentru acea sursă si construiti un cod cu lungimile codului optimal rezultat)
c. Este acest cod optimal, tot optimal si pentru sursa cu probabilitătile (1/5, 1/5, 1/5, 1/5, 1/5)? Expuneti rationamentul folosit. (*Indicatie*: Orice cod liber de prefix trebuie să satisfacă inegalitatea lui Kraft).

Problema 56.

Se consideră o variabilă aleatoare care ia echiprobabil m valori. Se stie că entropia acestei surse de informatie este $\log_2 m$ biti.

a. Pentru ce valori ale lui m lungimea medie a cuvântului de cod egalează entropia $H = \log_2 m$?

Solutie: Este necesar a fi satisfăcută egalitatea

$$\sum_{i=1}^{m} \frac{1}{m} l_i = \log_2 m$$

echivalentă succesiv cu

$$\frac{1}{m}\sum_{i=1}^{m} l_i = \log_2 m\,,\ \sum_{i=1}^{m} l_i = m\log_2 m\,,\ 2^{\sum_{i=1}^{m} l_i} = m^m\,.$$

Suma lungimilor cuvintelor de cod este totdeauna un număr întreg pozitiv. Ultima egalitate impune ca m să fie o putere întreagă si pozitivă a lui 2.

b. Se stie că pentru orice distributie probabilistică, lungimea medie a codului optim si entropia sunt în relatia $L < H + 1$. Se defineste ca redundanță a unui cod diferenta $\rho = L - H$. Pentru ce valori ale lui m, $2^k \le m \le 2^{k+1}$, redundanta codului optim este maximă? Care este valoarea limită a acestei redundante maxime, pentru $m \to \infty$?

Solutie: Pentru valorile extreme, 2^k si 2^{k+1}, redundantele sunt nule. Pentru valorile intermediare, cuvintele codului optim/compact au lugimile de k sau de $k + 1$ biti. Nu pot fi toate de lungime k deoarece cuvintele ar fi prea putine, codul n-ar mai fi unic decodabil. Pot fi toate de lungime $k + 1$ dar sunt prea multe cuvinte. După atribuire, sunt cuvinte ale codului diferite de unul din cuvintele în surplus prin numai un bit. Cuvintele de cod care au pereche în surplusul de cuvinte lungi de $k + 1$ biti si care diferă prin numai un bit de acelea pot fi scurtate cu acel bit si incluse în cod. Codul rămâne unic decodabil dar este si optimal (algoritmul Huffman poate confirma acest adevăr). După cum m excede limita 2^k cu 1, 2, …, $2^k - 1$ unităti, cuvintele de cod de lungime $k + 1$ sunt în număr de 2, 4, …, $2^{k+1} - 2$, adică $2(m - 2^k)$, iar cuvintele de cod de lungime k sunt în număr de $m - 2$, $m - 4$, …, $m - 2^{k+1} + 2$, adică $m - 2(m - 2^k) = 2^{k+1} - m$.

Lungimea medie a codului este

$$L = \frac{1}{m}\left((2^{k+1} - m)k + 2(m - 2^k)(k + 1)\right)$$

ceea ce după prelucrare devine

$$L = \frac{1}{m}\left(m(k + 2) - 2^{k+1}\right) = (k + 2) - \frac{2^{k+1}}{m}$$

Pentru $2^k < m < 2^{k+1}$, redundanta codului este

$$\rho = L - H = (k + 2) - \frac{2^{k+1}}{m} - \log_2 m$$

Functia

$$f(x) = (k+2) - \frac{2^{k+1}}{x} - \log_2 x$$

are un extrem (un maxim) în punctul de anulare a derivatei

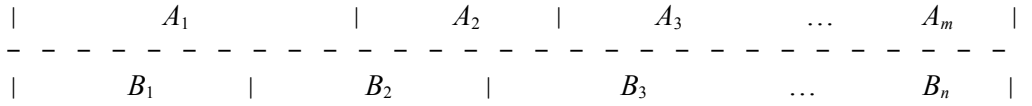
$$f'(x) = \frac{2^{k+1}}{x^2} - \frac{1}{x \ln 2}$$

adică pentru $x = 2^{k+1}\ln 2$. În acest punct functia $f(x)$ ia vloarea $1 - 1/\ln 2 - \ln\ln 2/\ln 2 \approx 0{,}0861$. Pentru $x \to \infty$ aceasta este si limita functiei în discutie. Aceeasi este si limita cerută de problemă deoarece un sir de valori m întregi, în particular diferite cu mai putin de o unitate de un x real nu poate conduce la altă limită.

Problema 57.

Un cod nu este unic decodabil dacă si numai dacă există o secventă finită de simboluri ale codului care poate fi interpretată în moduri diferite în secventele de cuvinte de cod.

În secvența



unde fiecare A_i si B_i sunt cuvinte de cod, pot apărea situatii de acest gen. Se observă că B_1 poate fi prefix pentru A_1, rezultând si un oarecare sufix reminiscent. Fiecare sufix reminiscent trebuie să fie la rându-i un prefix sau un cuvânt de cod, sau să aibă un alt cuvânt ca prefix propriu, ceea ce produce un alt sufix reminiscent. În final, ultimul sufix din secvență trebuie să fie un cuvânt de cod. Se poate pune la punct un test de unică decodabilitate (testul Sardinas-Patterson) în maniera care urmează:

- Se construieste o multime S care să cuprindă toate sufixele reminiscente posibile
- Codul este unic decodabil dacă si numai dacă mulțimea S nu contine cuvinte de cod.

a. Se admite că lungimile cuvintelor de cod sunt l_i, $i = 1, 2, \ldots, m$. Găsiti o limitare superioară pentru numărul de elemente din S
b. Determinati care din codurile următoare sunt unic decodabile:
   i. {0, 10, 11}
   ii. {0, 01, 11}
   iii. {0, 01, 10}
   iv. {0, 01}
   v. {00, 01, 10, 11}
   vi. {110, 11, 10}
   vii. {110, 11, 100, 00, 10}
c. Pentru fiecare din codurile unic decodabile de mai sus, construiti, dacă este posibil, o secventă infinită codată cu un punct de început cunoscut, astfel

încât ea să poată fi interpretată în două moduri. Aceasta arată că dacă cuvintele de cod finite sunt unic decodabile, cuvintele de cod infinite pot să nu fie unic decodabile.

d. Dovediti că o asemenea secventă nu poate apărea într-un cod liber de prefix.

Problema 58. Codul Shannon

Se consideră metoda următoare de a genera un cod pentru o variabilă aleatoare X care ia m valori $\{1, 2, \ldots, m\}$ cu probabilitătile $p_1, p_2, \ldots, p_m$. Se presupune că probabilitătile sunt ordonate astfel $p_1 \geq p_2 \geq \ldots \geq p_m$. Se defineste

$$F_i = \sum_{k=1}^{i-1} p_k$$

suma probabilitătilor tuturor simbolurilor de indice mai mic decât i. Cuvântul de cod asociat lui i este numărul $F_i \in [0,1]$ rotunjit la l_i biti cu l_i partea întreagă a numărului $\log_2(1/p_i)$.

- Arătati că un cod construit astfel este liber de prefix si lungimea lui medie satisface relatia

$$H(X) \leq L < H(X) + 1$$

- Construiti codul pentru cazul în care probabilitătile sunt {0,5 0,25 0,125 0,125}.

Problema 59. Compresia optimă simplă pentru o sursă Markov

Se consideră procesul Markov $U_1, U_2, \ldots$, un proces cu trei stări care are matricea de tranzitie

$U_{n-1} \backslash U_n$	S_1	S_2	S_3
S_1	1/2	1/4	1/4
S_2	1/4	1/2	1/4
S_3	0	1/2	1/2

cu probabilitatea tranzitiei din starea S_3 în starea S_1 nulă. Proiectati trei coduri C_1, C_2, C_3 (unul pentru fiecare stare S_1, S_2, S_3). Fiecare cod aplică elemente din multimea $\{S_1, S_2, S_3\}$ pe secvente de valori binare 0 si 1, astfel încât acest proces Markov să poată fi transmis cu compresie maximă prin schema următoare:

a. Se observă simbolul prezent S_i
b. Se selectează codul C_i
c. Se observă simbolul următor S_j si se trimite cuvântul de cod C_j care corespunde lui S_j
d. Se repetă operatia pentru simbolul următor.

Care este lungimea medie a mesajului asociat simbolului următor conditionată de starea anterioară $S = S_i$ utilizând această schemă de codare?

Care este numărul mediu neconditionat de biti pe simbol al sursei?

Puneti în relatie această lungime cu rata entropiei $H(U)$ a lantului Markov.
Solutie: Starea stationară a procesului este descrisă de vectorul de probabilităti [2/9 4/9 3/9] si entropia ei în absenta vreunei conditionări între simboluri este 1,5305. Cu matricea de tranzitie dată, entropiile pentru fiecare din cele trei stări ale procesului ca stare de plecare sunt respectiv 3/2, 3/2, 1 biti. Entropia medie este

$$[2/9 \;\; 4/9 \;\; 3/9]\cdot[3/2 \;\; 3/2 \;\; 1]^T = 4/3 = 1{,}3333$$

mai mică decât cea a stării stationare, asa cum era de asteptat.
Codurile binare separate C_1, C_2, C_3, optime pentru fiecare stare ca stare de plecare duc la lungimi medii egale cu entropiile în primele două cazuri deoarece tranzitiile respective au toate caracteristicile unor surse de informatie speciale (probabilitătile sunt puteri întregi ale lui 2). Tabelul alăturat contine codurile compacte pentru fiecare stare de plecare.

$S_1 \rightarrow$	Pr.	C_1	$S_2 \rightarrow$	Pr.	C_2	$S_3 \rightarrow$	Pr.	C_3
S_1	1/2	0	S_1	1/4	10	S_1	0	11
S_2	1/4	10	S_2	1/2	0	S_2	1/2	0
S_3	1/4	11	S_3	1/4	11	S_3	1/2	10

Lungimile medii ale acestor trei coduri sunt toate egale cu 1,5 biti/simbol.
Procesul Markov din enunt poate fi tratat si ca o succesiune de tranzitii, în număr de 9, ($S_i \rightarrow S_j$), $i, j = 1, 2, 3$, cu probabilitătile [1/9 1/18 1/18 1/9 2/9 1/9 0 1/6 1/6]. Entropia este 2,8638 si este egală cu suma entropiilor evaluate mai devreme.
Rezultatul codării Huffman al acestei surse definită prin tranzitiile ei si probabilitătile asociate este dat în tabel.
Lungimea medie a codului este 2,9444 bit/simbol, întrucâtva mai mare decât entropia.

$S_i \rightarrow S_j$	Probabilităti	Cod Huffman
$(i, j) = (1, 1)$	1/9	110
$(i, j) = (1, 2)$	1/18	1010
$(i, j) = (1, 3)$	1/18	10110
$(i, j) = (2, 1)$	1/9	111
$(i, j) = (2, 2)$	2/9	01
$(i, j) = (2, 3)$	1/9	100
$(i, j) = (3, 1)$	0	10111
$(i, j) = (3, 2)$	1/6	000
$(i, j) = (3, 3)$	1/6	001

Codarea după schema propusă în enunț conduce la o lungime medie de 3 biti/pereche-de-simboluri si este mai putin eficientă decât aceea dată în tabelul al doilea.

Problema 60: Algoritmi speciali

Codarea prin metoda *run-length* este una din cele mai simple sau poate cea mai simplă metodă de codare în scopul comprimării fără pierdere de informatie a unui fisier care contine o secventă de întregi.

a. Scrieti o procedură simplă (eventual în pseudocod) care primeste o secventă de numere naturale a_1, …, a_n si returnează o secventă codată de întregi b_1, …, b_m. Ideea centrală constă în a genera secventa de iesire astfel ca o subsecventă din secventa de intrare lungă de 3 sau mai multe numere consecutive identice să fie codată cu o pereche de întregi: primul care indică lungimea secventei, al doilea care este însusi numărul repetat. De pildă, secventa 6, 2, 3, 4, 4, 4, 5, 6, 7, 7, 7, 7, 8 este codată ca 6, 2, 3, −3, 4, 5, 6, −4, 7, 8. După cum se observă, secventa codată este mai scurtă decât cea initială.

Solutie: Procedură (o posibilitate, în pseudocod):

```
citeste număr_nou;
contor=1;
număr_vechi=număr_nou;
repetă
      citeste număr_nou;
      dacă număr_nou=număr vechi
            contor=contor+1;
      altminteri
            dacă contor>1
                  scrie -contor, mumăr_vechi;
                  contor=1;
            altminteri
                  scrie număr_vechi;
            final_dacă;
      număr_vechi=număr_nou;
      final_dacă;
până când EOF
dacă contor>1
      scrie -contor, mumăr_vechi;
altminteri
      scrie număr_vechi;
final_dacă
```

b. De ce este necesar semnul minus?

Solutie: Semnul minus este marcajul distinctiv pentru frecventele numerelor care apar repetat în secventa de intrare.

c. Scrieti o procedură de decodare care primeste secventa codată si restituie secventa de dinainte de codare

Solutie: Procedură (în pseudocod):

```
cât_timp ¬EOF
      citeste cod;
      dacă cod<0
```

```
            n1= -cod;
            citeste cod;
            scrie (de n1 ori) cod;
      altminteri
            scrie cod;
      final_dacă
final_cât_timp
```

d. Descrieti tipul de secvente la intrare care produce cel mai rău caz de complexitate temporală pentru combinatia de algoritmi de codare si de decodare (numărati numai operatiile de comparare)

Solutie: Dacă secventa de numere care este supusă codării este de lungime n, atunci comparatia între **număr_nou** si **număr-vechi** se execută totdeauna de $n - 1$ ori. Comparatia **contor > 1** se execută numai atunci când **număr_nou** si **număr-vechi** sunt diferite. Dacă secventa dată nu contine nici o succesiune de 2 sau mai multe numere identice atunci comparatia se execută tot de $n - 1$ ori si acesta este cazul cel mai nefericit. În finalul operatiei de codare se mai execută o comparatie **contor > 1** ceea ce aduce totalul la $2n - 1$.

Operatia de decodare execută comparatia **cod < 0** o dată la citirea primului **cod** din cele în total m, $m \leq n$. De fiecare dată când citeste un **cod** negativ, urmează citirea unui **cod** pozitiv si comparatia **cod < 0** este repetată după scrierea unei secvente de lungimea cuvenită, secventă de numere naturale identice. Cazul cel mai rău este cel în care valorile **cod** citite succesiv sunt toate pozitive. Atunci $m = n$ si numărul comparatiilor este $1 + (m - 1) = m = n$.

Codarea si decodarea utilizează în acest caz, cel mai rău posibil, $3n - 1$ comparatii.

e. Produceti o formulă pentru fiecare algoritm, care să exprime numărul de comparatii necesare ca functie de lungimea secventei de intrare n în cazul cel mai rău

Solutie: Formulele sunt date la punctul anterior.

f. Utilizati notatia "big-O" pentru a descrie cazul cel mai rău pentru algoritmii utilizati.

Solutie: $O(n)$.

CRIPTAREA

Lucrarea 7.

Tema 1: Criptarea RSA

Criptografia urmează scenarii de genul următor: două persoane **A** si **B** doresc să comunice desi simt în mediul de comunicare prezenta unei alte persoane deosebit de curioase **E**. Se presupune că **A** vrea să trimită lui **B** mesajul x. Criptografia oferă solutii în forma următoare: **A** calculează o functie de x, $e(x)$, utilizând o cheie secretă si trimite $e(x)$ pe canalul ascultat si de **E**. **B** primeste mesajul criptat $e(x)$ si utilizând cheia sa (care în criptografia traditională este aceeasi cu cheia lui **A**, dar în criptografia modernă nu este aceeasi) calculează o functie $d(e(x)) = x$, recuperând astfel mesajul. Se prezumă că **E** este incapabil să recupereze pe x din $e(x)$ deoarece nu are cheia necesară.
O metodă criptografică clasică este *substituirea de litere*. **A** si **B** cad de acord asupra unei permutări π a literelor A, …, Z si un mesaj $m = a_1a_2\ldots a_n$ alcătuit din n litere devine $e(m) = \pi(a_1)\pi(a_2)\ldots\pi(a_n)$. În pofida faptului că există cel putin 26! chei posibile (cu referire la limba engleză), acesta este un sistem de criptare foarte slab: el este expus la *atacul* evident prin *frecventa literelor*.

Cu concursul calculatoarelor, criptografii pot crea sisteme de criptare în care sunt substituite *blocuri* întregi *de litere* (sau de biti), rezistente astfel la atacul prin frecvente. Cel mai de succes dintre ele a fost si încă este Data Encryption Standard (DES), o metodă de criptare acceptată de guvernul american, propusă de IBM în 1976. DES utilizează o cheie pe 64 de biti (pentru codarea fisierelor mai lungi, acestea se sparg în blocuri de 8 biti). Cei care au propus DES pretind că este un sistem sigur; sunt si detractori care-l suspectează că la modul subtil nu este atât de sigur. Dimensiunea cheii de 56 de biti este în general considerată inadecvată pentru criptografia serioasă, deoarece 2^{56} nu mai este un număr mare.

Cam la aceeasi vreme când DES a fost inventat, a fost propusă o idee nouă foarte atractivă: criptografie cu cheie publică (*public-key cryptography*). În comunicare, persoana **B** ar avea două chei: cheia sa privată k_d, cunoscută numai de el si o a doua cheie publică, k_c. Cheia k_c este cunoscută tuturor – este de pildă pe pagina gazdă a lui **B**. Cu o metaforă mai curând dubioasă, este ca si cum ar exista mai multe cópii ale unui lacăt cu care oamenii pot închide cutii care contin mesaje si sunt trimise lui **B**. Dacă cineva, de pilă **A** doreste să trimită un mesaj x lui **B**, atunci îl criptează sub forma $e(x)$ care utilizează cheia publică. **B** îl decriptează folosindu-se de cheia lui privată. Partea inteligentă a sistemului este că: (1) decriptarea este corectă, adică $d(e(x)) = x$; (2) nu se poate calcula într-un timp rezonabil cheia privată a lui **B** din cheia lui publică si nici x din $e(x)$, astfel că protocolul este sigur, cam la fel cum nu există o cale de a imagina cheia pentru un lacăt bun.

Sistemul RSA – de la intialele a trei cercetători (Ron Rivest, Adi Shamir si Len Adleman) care l-au inventat – este un mod inteligent de a realiza ideea de cheie publică. Iată cum lucrează aceasta:

- *Generarea cheii*. **B** selectează două numere prime mari, p si q. În zilele noastre, "mare" înseamnă câteva sute de digiti zecimali, curând poate va însemna peste o mie. Pentru a găsi asemenea numere, **B** generează repetat intregi din această gamă si îi supune unui test de primalitate datorat lui Fermat, până când două din ele trec testul. Apoi **B** calculează $n = p.q$. De asemenea, **B** generează la întâmplare un întreg $e < n$, cu singura restrictie de a fi prim cu $(p - 1)$ si cu $(q - 1)$. Perechea (n, e) este de acum *cheia publică* a lui **B** si o face cunoscută tuturor. Practic, pentru a face codarea mai usoară, e este luat adesea 3. Desigur, trebuie evitate numerele prime care sunt din clasa 1 mod 3.

  Acum **B** poate genera cheia lui secretă. Tot ce are de făcut este a calcula prin mijlocirea algoritmului lui Euclid numărul $d = e^{-1} \bmod (p - 1)(q - 1)$. Perechea (n, d) este *cheia privată* a lui **B**.
- *Operarea*. Ori de câte ori **A** sau oricine altcineva doreste să-i trimită un mesaj lui **B** procedează astfel:
  - Fragmentează mesajul în siruri de biti de lungime $\lfloor \log n \rfloor$ obtinută prin operatia $\lfloor \cdot \rfloor$ de trunchiere a unui număr real. Si aici este vorba

de mai multe sute de biti. Se codează fiecare sir de biti prin algoritmul care urmează.

- Fie x un astfel de sir de biti si se consideră ca fiind un întreg mod n. **A** calculează x^e mod n. Acesta aste mesajul codat $e(x)$ pe care **A** îl trimite lui **B**.
- La primirea lui $e(x)$, **B** calculează $e(x)^d$ mod n. Putină algebră care apelează la mica teoremă a lui Fermat pentru produsul a două numere prime produce:

$$e(x)^d = x^{d.e} = x^{1 + m(p-1)(q-1)} = x \bmod n$$

De observat că această secventă de egalităti probează faptul că **B** decriptează corect mesajul primit. Prima egalitate tine seamă de definitia lui $e(x)$. A doua decurge din faptul că d este inversul lui e mod $(p-1)(q-1)$ si astfel, dacă se multiplică d cu e se obtine 1 plus un multiplu de $(p-1)(q-1)$. Ultima egalitate aminteste de mica teoremă a lui Fermat: $x^{(p-1)(q-1)} = 1 \bmod n$, cu exceptia cazului în care x este un multiplu de p sau de q, o posibilitate foarte putin probabilă; astfel, multiplul de $(p-1)(q-1)$ din exponent se poate ignora.

Astfel **A** poate coda folosind cheia publică a lui **B**. **B** poate decoda utilizând cheia privată pe care numai el o cunoaste. Dar ce se poate spune despre "curiosi"? Dacă **E** preia $e(x) = x^e$ mod n, el poate face mai multe lucruri. Poate încerca toti x-ii posibili, să-i codeze cu cheia publică a lui **B** si să găsească x-ul corect – dar asta ia mult prea mult timp. Sau, poate să calculeze $d = e^{-1}$ mod $(p-1)(q-1)$ si din asta $e(x)^d$ mod n care este x. Dar pentru a face asta trebuie să stie numărul $(p-1)(q-1)$; si dacă stie atât $p.q = n$ cât si $(p-1)(q-1)$ stie p si q. Cu alte cuvinte, **E** ar putea factoriza pe n, un produs arbitrar de două numere prime mari, dar nimeni nu stie cum să facă asta rapid. Sau, în sfârsit, **E** si-ar putea utiliza propria metodă ingenioasă pentru a decoda $e(x)$ fără factorizarea lui n, dar este o convingere largă că nu există o asemenea metodă.

Astfel, după toate dovezile existente, RSA este sigur pentru un n suficient de mare.

Programul alăturat este o ilustrare a modului cum lucrează sistemul de criptare RSA. Operatiile de criptare si de decriptare sunt puse laolaltă, în sucesiunea firească. Nu s-a recurs la numere prime p, q foarte mari pentru a nu depăsi posibilitătile aritmetice curente ale pachetului Matlab. Studentii sunt îndemnati să înlocuiască numerele prime p, q propuse cu altele. Sistemul de criptare/decriptare va functiona atât timp cât numerele alese îndeplinesc conditiile precizate mai sus. Se recomandă si încercarea unor numere p, q compuse. Se va putea observa în acest caz functionarea defectuoasă sau nefunctionarea sistemului.

```
% Program de criptare/decriptare cu cheie publica RSA
clear
e=5;    % Prima parte a cheii publice
prime=primes(1000); % Se genereaza numere prime < 1000
[i,m1]=size(prime);
pri=zeros(1,m1);
k=0;
```

```
for i=4:m1
    if mod(prime(i)-1,e)~=0
        k=k+1;
        pri(k)=prime(i);
    end
end
% Se retin numai numerele prime cu proprietatea "e nu divide pe
prim-1"
m1=k;
i2=max([1 1],floor(m1*rand(1,2)))
% Se retin la intamplare doua numere prime p si q
p=pri(i2(1));
q=pri(i2(2));
p_si_q=[p q]
n=p*q;
cheia_publica=[e n]    % Se formeaza cheia publica: (e, n)
m=(p-1)*(q-1);
d=1;b=1;
while mod(d*e,m)~=1
    d=d+1;
end
% Se stabileste partea secreta a cheii private
cheia_privata=[d n]
m=floor(n*rand);   % Se genereaza mesajul de criptat (generat
aleator)
mesajul_de_criptat=m
b=1;
for i=1:e
    b=m*b;
    a=floor(b/n);
    b=b-a*n;
end        % Se executa operatia de criptare
m=b;       % Se afiseaza mesajul criptat
mesajul_criptat=b
cheia_privata=[d n]
b=1;
for i=1:d
    b=m*b;
    a=floor(b/n);
    b=b-a*n;
end        % Se executa operatia de decriptare
mesajul_decriptat=b     % Se afiseaza mesajul decriptat
```

Tema 2: Criptarea cu curbe eliptice

Curbe eliptice peste Z_p: Fiind dat un număr prim p, o curbă eliptică peste Z_p este o congruență $y^2 = x^3 + ax + b \bmod p$ la care se adaugă un punct O la infinit astfel încât să fie satisfăcută o conditie de "nesingularitate" $4a^3 + 27b^2 \neq 0 \bmod p$.

Fiind date punctele $P_1 = (x_1, y_1)$ si $P_2 = (x_2, y_2)$, se defineste pe curba eliptică

$$P_1 + P_2 = (\lambda^2 - x_1 - x_2,\ \lambda(x_1 - x_3) - y_1)$$

cu

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & pentru \quad P_1 \neq P_2 \\ \dfrac{3x_1^2 + a}{2y_1} & pentru \quad P_1 = P_2 \end{cases}$$

Se consideră totodată că $(x_1, y_1) + (x_1, -y_1) = O$ si $P_1 + O = O + P_1 = P_1$. Acesta este un grup abelian denumit $E_p(a, b)$.

Exemplu: Se consideră curba eliptică $y^2 = x^3 + x + 6$ peste Z_{11}^*. Pentru a determina punctele din E, se ia fiecare punct din Z_{11}, se calculează $x^3 + x + 6$ si apoi se rezolvă ecuatia $y^2 \equiv x^3 + x + 6 \bmod 11$.

Calculul acesta cere evaluarea de radicali *modulo* 11. Există o formulă explicită pentru aceasta deoarece $11 \equiv 3 \bmod 4$. De fapt, rădăcinile pătrate ale unui rest pătratic r sunt $\pm r^{(11+1)/4} \equiv r^3 \bmod 11$. Prin calcul si pentru calcule ulterioare se completează tabelul de mai jos.

Curba eliptică are pe ea 13 puncte (x, y), dacă se include si punctul de la infinit. Fiind de ordinul 13 (prim) ea însăsi trebuie să fie un grup ciclic. De pildă, dacă se ia generatorul $\alpha = (2, 7)$, puterile lui α, de pildă $(2, 7) + (2, 7)$, pot fi calculate după cum urmează:

x	$x^3 + x + 6$	$\in QR_{11}$?	y
0	6	nu	
1	8	nu	
2	5	da	4, 7
3	3	da	5, 6
4	8	nu	
5	4	da	2, 9
6	8	nu	
7	4	da	2, 9
8	9	da	3, 8
9	7	nu	
10	4	da	2, 9

Se evaluează mai întâi λ:

$$\lambda = (3 \cdot 2^2 + 1)(2 \cdot 7)^{-1} \bmod 11 = (2 \cdot 3)^{-1} \bmod 11 = 2 \cdot 4 \bmod 11 = 8$$

Asadar $x_3 = 8^2 - 2 - 2 = 5 \bmod 11$ si $y_3 = 8(2 - 5) - 7 = 2 \bmod 11$. Decurge de aici că $(2, 7) + (2, 7) = (5, 2)$.

Apoi, $(2, 7) + (2, 7) + (2, 7) = 2(2, 7) + (2, 7) = (5, 2) + (2, 7)$.

Sisteme de criptare cu curbe eliptice: Algoritmul foarte general ElGamal se poate aplica la subgrupul ciclic Z_{n_1} al lui $E_p(a, b)$, dar factorul lui de dezvoltare este 4 (fată de 2 pentru Z_p). În plus, spatiul textelor în clar (*plaintext space*) constă din punctele din $E_p(a, b)$ si nu există metodă convenabilă de a genera în mod determinist puncte din $E_p(a, b)$.

Se ia o curbă eliptică E_p care contine un subgrup ciclic $H = Z_{n_1}$ cu algoritm discret intractabil. Spatiul textelor în clar este $Z_p^* \times Z_p^*$ si spatiul textelor cifrate (*ciphertext*) este $E_p \times Z_p^* \times Z_p^*$.

Sistemul Menezes – Vanstone bazat pe curbe eliptice:

Public: $\alpha, \beta \in E_p$

Privat: a astfel ca $\beta = a\alpha$

Criptare: Se alege aleator $k \in Z_{n_1}$
$(x, k) \rightarrow E(x, k) = (y_0, y_1, y_2)$:
$y_0 = k\alpha$, $(c_1, c_2) = k\beta$
$x = (x_1, x_2)$, $y_1 \equiv c_1 x_1 \bmod p$ şi $y_2 \equiv c_2 x_2 \bmod p$

Decriptare: $(y_0, y_1, y_2) \rightarrow D(y_0, y_1, y_2) = \left(y_1 c_1^{-1} \bmod p, y_2 c_2^{-1} \bmod p\right)$ cu
$ay_0 = (c_1, c_2)$.

Se propune spre studiu si experimentare programul Matlab alăturat.

```
% Program de criptare/decriptare cu curbe eliptice
clear
prime=primes(100); % generare de numere prime
[m,n]=size(prime);
m=unidrnd(n,1,1);
p=prime(m)
p=11
a=1;b=6;
if mod(4*a^3+27*b^2,p)==0
    mesaj='Nu merge!'
    break
end
x=1:(p-1);
x2=mod(x.^2,p);
y2=mod(x.^3+a*x+b,p);
k=0;
for i=1:(p-1)
    for j=1:(p-1)
        if y2(i)==x2(j)
            k=k+1;
            per(k,1:2)=[i j]; % calculul punctelor de pe curba
eliptica
        end
    end
end
[m,n]=size(per)
n=unidrnd(m,1,1);
n=6
alfa=per(n,:) % alegere dintre punctele de pe curba eliptica
alfa=[2 7]
for i=1:(p-1)
    for j=1:(p-1)
        if mod(x(i)*x(j),p)==1
            invx(i)=x(j);
```

```
        end
    end
end
exs=unidrnd(m-1,1,1) % alegere exponent secret
exs=7
la=mod(2*alfa(2),p)
la=invx(la)
la=mod((3*alfa(1)^2+exs)*la,p)
beta=[0 0];
for i=1:la
    beta(1)=mod(la^2-alfa(1)-beta(1),p);
    beta(2)=mod(la*(alfa(1)-beta(1))-alfa(2),p);
end
beta
```

Problema 61. Sistem RSA incorect

Se presupune că din eroare sau deliberat cineva alege o valoare pentru n care nu este un produs de două numere prime, adică $n = pq$ cu $p > 1$, $q > 1$ si q este compus. Ar fi, evident, mai usor de factorizat ceea ce ar pune o problemă de risc sub aspectul securitătii. Dar vor functiona operatiile de criptare si de decriptare cu acest n? Sustineti răspunsul.

Solutie: Nu, criptarea/decriptarea nu mai functionează. Pentru a avea un contraexemplu, fie $n = 45 = 5.9$. Dacă punem $p = 5$ si $q = 9$, atunci pentru o cheie de criptare $(e, 45)$, am alege o cheie de decriptare $(d, 45)$ pentru care $de \equiv 1 \bmod ((5 – 1)(9 – 1))$, adică

$$de \equiv 1 \pmod{32}$$

Astfel, dacă $e = 5$, inversul mod 32 este $d = 13$. Dar dacă criptăm si decriptăm mesajul $M = 2$, obtinem

$$(2^5)^{13} = 2^{65} \equiv 32 \pmod{45}.$$

Mai general, RSA se bazează pe faptul că $(M^e)^d \equiv 1 \pmod{n}$, deoarece această relatie este cea care asigură că decriptarea este inversa criptării. Relatia se va mentine dacă $de \equiv 1 \pmod{\varphi(n)}$. Dar, dacă q este compus, functia totient $\varphi(n) \neq (p – 1)(q – 1)$ si astfel vom avea propozitia $de \equiv 1 \bmod ((p – 1)(q – 1))$ falsă. În consecintă, în cazul exemplificat criptarea RSA nu poate lucra.

Problema 62. Spargerea unui sistem RSA

Fie p si q numere prime si $N = pq$. Arătati cum se pot determina p si q fiind date N si $(p – 1)(q – 1)$. Cu alte cuvinte, fiind dată cheia publică (e, N), e exponentul de criptare si N modulul RSA, precum si valoarea $\varphi(N) = (p – 1)(q – 1)$, este posibil a calcula p si q prin operatii algebrice simple (în timp polinomial). Aceasta arată că determinarea lui $\varphi(N)$ este la fel de grea ca si factorizarea.

Solutie: Fiind date $pq = N$ si $(p – 1)(q – 1) = \varphi(N)$, urmează a afla numerele p si q. Se poate pune $q = N/p$ după care o înlocuire în ecuatia a doua produce după câteva prelucrări algebrice ecuatia în p

$$p^2 + (\varphi(N) – N – 1)p + N = 0$$

care poate fi rezolvată aplicând formula cunoscută. Dacă p a fost găsit, este usor a-l găsi si pe q.

Problema 63. O problemă de criptare cu cadre didactice

a. Se presupune că la o anumită disciplină activează trei profesori si doi asistenti. Solutiile la tema de casă următoare sunt criptate printr-o cheie de criptare în uzul tuturor celor cinci. Cei trei profesori, sau un asistent si un profesor, sau ambii asistenti trebuie să poată accesa solutiile. Sugerati o schemă de împărtire a secretului care să asigure aceste cerinte (*Indicatie*: Încercati ponderi.).

Solutie: Utilizati o schemă partajată 3-din-7 pentru a genera 7 părti ale acestui secret. Dati o parte fiecărui profesor, si două părti fiecărui asistent. Cei trei profesori laolaltă au 3 părti; un profesor si un asistent au împreună 3 părti si cei doi asistenti au 4 părti; astfel, în orice caz, aceste subseturi au părti suficiente pentru a recupera secretul.

Mai precis, imaginati-vă secretul ca ordonata la origine a unui polinom de gradul al doilea modular $f(x)$. Fiecărui profesor i se dă câte un punct pe $f(x)$ si fiecărui asistent câte două puncte pe curba $f(x)$, punctele fiind diferite în perechi. Pentru a descifra $f(x)$ si ordonata ei la origine sunt necesare cel putin trei puncte. Acest minim de puncte poate fi cumulat numai de trei profesori, de un profesor si un asistent si de cei doi asistenti. Acelasi minim nu poate fi întrunit numai de doi profesori sau numai de un asistent.

b. Se presupune acum că grupa de cursanti este instruită de trei profesori, fiecare cu doi asistenti proprii. Oricare doi profesori pot accesa datele atât timp cât unul din asistentii fiecărui profesor (un total de cel putin patru persoane) este de asemenea prezent. Acum cum se procedează?

Solutie: Utilizati un sistem de partajare a secretului 2-din-3 pentru a genera părtile s_1, s_2, s_3 ale secretului. Apoi, utilizati o schemă de partajare 4-din-5 pentru a genera sub-părtile $s_{1,1}$, …, $s_{1,5}$ ale lui s_1, tratând s_1 ca secretul (unic); se dau trei sub-părti primului profesor si o subparte fiecăruia din asistentii lui. Se procedează analog cu ceilalti profesori.

Iată o descriere mai exactă. Mai întâi se ia la întâmplare un polinom de gradul 1, $P(x)$, astfel încât secretul este $P(0)$. Apoi tot la întâmplare se iau trei polinoame de gradul 3, $Q_1(x)$, $Q_2(x)$ si $Q_3(x)$, astfel încât $Q_i(0) = P(i)$ pentru oricare $i = 1, 2, 3$. Cu alte cuvinte, secretul fiecărui polinom $Q_i(x)$ este una din părtile derivate din partajarea secretului bazat pe $P(x)$.

Acum pentru fiecare $Q_i(x)$ generăm 5 sub-părti $Q_i(1)$, …, $Q_i(5)$, atribuind 3 sub-părti profesorului si câte una asistentilor lui. Atunci, pentru a recupera mesajul, fiecare profesor i se întruneste cu unul din asistentii săi si recuperează secretul polinomului $Q_i(x)$, care este partea lui din secretul polinomului $P(x)$. În final, doi profesori se pot întruni pentru a recupera $P(0)$.

CODAREA PENTRU CANALE CU PERTURBATII

Lucrarea 8

Tema 1: Codurile Hamming (corectoare de o eroare)

Codurile Hamming în forma lor originară au cuvintele de lungime $n = 2^m - 1$ cu $m = 3, 4, 5, \ldots$ Acestea sunt codurile care sunt propuse spre studiu în această lucrare. Prin urmare, este vorba aici de coduri Hamming (7,4), (15,11), (31,26) etc., cu primul număr lungimea n a cuvântului de cod, cu numărul al doilea lungimea $n - m$ a părtii purtătoare de informatie, de regulă separată si usor separabilă de cei m biti de protectie, biti de paritate, cum mai sunt denumiti oarecum impropriu.
În linia a doua a programului Matlab alăturat, se face atribuirea pentru m a unei valori naturale mai mari decât 2. Este singura modificare necesară pentru a genera coduri Hamming diverse si a le studia functionarea. Programul suportă valori variate pentru m, dar nu se recomandă depăsirea valorii 5 deoarece cazurile (7,4), (15,11), (31,26) sunt suficient de concludente si urmărirea calculelor pe monitor este mult usurată.

La apelarea programului **hamm**, se generează mai întâi matricea de control-verificare a codului. Aceasta se obtine foarte simplu: este rezultatul numărării în binar de la 1 la n.
Pentru a face codul separabil (partea informatională a cuvintelor codului separată de partea redundantă, de protectie) este necesară o ordonare a matricei de verificare. Ordonarea deplasează coloanele care contin o singură unitate binară la una din extremitătile matricei – aici la extremitatea din dreapta – pentru a constitui acolo o (sub)matrice unitate $m \times m$. Submatricea $m \times (n - m)$ rămasă devine după transpunere matricea generatoare a bitilor de control (numiti, cum s-a spus, si biti de paritate). Aceasta, alăturată unei (sub)matrici unitate $(n - m) \times (n - m)$ conduce la matricea generatoare a codului.
Odată stabilite aceste matrici specifice codului, se generează aleator un cuvânt informational de lungime $n - m$ si se face codarea lui. Pentru cuvântul de cod rezultat se calculează sindromul. Cum este de asteptat, vectorul sindrom are toate componentele nule.
Se inversează deliberat unul din bitii cuvântului de cod. Această schimbare este simularea aparitiei unei erori în cuvântul transmis si receptionat astfel, cu eroare. Sindromul va fi de această dată diferit de vectorul nul asociat exclusiv cuvintelor receptionate care apartin codului. Se observă că vectorul simdrom se potriveste pe una din coloanele matricei de verificare. Pozitia coloanei este aceeasi cu pozitia bitului care trebuie corectat.
Se recomandă executarea repetată a programului cu acelasi m sau cu valori m diferite.
Sunt încurajate modificările constructive ale programului propus.

```
% Program hamm pentru codurile Hamming
clear
m=4;
n=2^m-1;
s=strcat('Codul Hamming (',int2str(n),',',int2str(n-m),')');
disp(' ')
disp(s)
for i=1:n
    a=dec2binvec(i,m);
    h1(:,i)=a';
end    % generarea matricii de control
h=h1;
matricea_de_control=h
z=sum(h);
j=0;k=0;
for i=1:n
    if z(i)>1
        j=j+1;
        h(:,j)=h1(:,i);
    else
        k=k+1;
        h(:,n-m+k)=h1(:,i);
    end
end    % ordonarea matricii de control
```

```
matricea_de_control_ordonata=h     % se afiseaza matricea de
control ordonata
g=[eye(n-m) (h(:,1:(n-m)))'];    % construirea matricei
generatoare
matricea_generatoare=g'
for i=1:n
    z(i)=binvec2dec(h(:,i)');
end
inf=unidrnd(2,1,n-m)-1;   % se genereaza n-m biti
informaationali
biti_de_informatie=inf    % se afiseaza bitii informationali
for j=1:m
    p(j)=0;
    for i=1:n-m
        p(j)=bitxor(p(j),bitand(h(j,i),inf(i)));
    end
end    % se calculeaza cei m biti de protectie
cod=[inf p];    % se genereaza cuvantul de cod
cuvantul_de_cod=cod
b=h*cod';
b=b-2*(floor(b/2));
vectorul_sindrom=b    % se afiseaza sindromul
l=unidrnd(n);    % se introduce eroare pe bitul l
s=strcat('Se inverseaza bitul de pe pozitia');
disp(s)
disp(int2str(l))
cod(l)=bitxor(1,cod(l));
cuvantul_cu_eroare=cod
b=h*cod';
b=b-2*(floor(b/2));
vectorul_sindrom=b    % se afiseaza sindromul
```

Tema 2: Un cod Hamming (7,4) în acțiune

Programul acesta reproduce în bună măsură programul precedent. Este limitat numai la codul Hamming (7,4).
Programul codează un text relativ scurt, în jur 60 de caractere. După codare, unul din cuvintele de cod este modificat: un bit este inversat. Se decodează sirul de biti receptionat cu eroare, textul se afisează, după o decodare simplă prin separarea părtii de informatie, 4 biti, de surplusul de trei biti, 7 – 4 = 3, adăugati pentru protectie. Dacă bitul modificat este în zona informațională, efectul erorii poate fi imediat observat. Dacă bitul eronat este pe una din pozitiile 5, 6 sau 7 în cuvântul receptionat, eroarea nu se observă la afisaj desi ea există.
În faza ultimă, bitul eronat este localizat si corectat. Textul afisat după decodare si inversarea bitului eronat este din nou corect.
Se recomandă repetarea executiei programului, eventual pentru alte siruri de caractere de codat/transmis.

```
% Program cod_dec_hamm pentru codare-decodare Hamming (7,4)
clear
m=3;    % a nu se modifica
```

```
n=2^m-1;
disp(' ')
sir='Transmiterea si codarea informatiei: o disciplina
prietenoasa';
disp('Se codeaza textul:')
disp(' ')
disp(sir)
s=strcat('Se utilizeaza codul Hamming
(',int2str(n),',',int2str(n-m),')');
disp(' ')
disp(s)
for i=1:n
    a=dec2binvec(i,m);
    h1(:,i)=a';
end    % generarea matricii de control
h=h1;
z=sum(h);
j=0;k=0;
for i=1:n
    if z(i)>1
        j=j+1;
        h(:,j)=h1(:,i);
    else
        k=k+1;
        h(:,n-m+k)=h1(:,i);
    end
end    % ordonarea matricii de control
g=[eye(n-m) (h(:,1:(n-m)))'];    % construirea matricei
generatoare
aski=double(sir);    % codurile ASCII ale caracterelor din sir
[i,nc]=size(aski);    % nc este numarul de caractere din sir
disp(' ')
s=strcat('Numar de caractere:',num2str(nc));
disp(s)
biti=zeros(1,14*nc);
for i=1:nc
    vec=dec2binvec(aski(i),8);
    prov=zeros(n-m:n);
    for i1=1:n
        prov(:,i1)=and(vec(1:4),(g(:,i1))');
    end
    pro=zeros(1,n);
    for i1=1:n
        for j=1:(n-m)
            pro(i1)=xor(pro(i1),prov(j,i1));
        end
    end
    biti(14*(i-1)+1:14*i-7)=pro;
    prov=zeros(n-m:n);
    for i1=1:n
        prov(:,i1)=and(vec(5:8),(g(:,i1))');
    end
    pro=zeros(1,n);
    for i1=1:n
```

```
        for j=1:(n-m)
            pro(i1)=xor(pro(i1),prov(j,i1));
        end
    end
    biti(14*(i-1)+8:14*i)=pro;
end    % se compun cuvintele de cod Hamming
k1=unidrnd(61);    % se alege aleator cuvantul de cod purtator
de eroare
k2=unidrnd(7);    % se alege aleator pozitia bitului eronat
s=strcat('Se introduce o eroare în cuvantul de
cod:',num2str(k1),' bitul:',num2str(k2));
disp(' ')
disp(s)
biti(14*(k1-1)+k2)=xor(1,biti(14*(k1-1)+k2));    % se inverseaza
un bit
b=zeros(1,nc);
for i=1:nc
    a1=[biti(14*(i-1)+1:14*i-7-3) biti(14*(i-1)+8:14*i-3)];
    b(i)=binvec2dec(a1);
end
disp(' ')
disp('Textul cu eroarea (eventual observabila):')
disp(' ')
disp(char(b))
for i=1:2*nc
    cuv=(biti(7*(i-1)+1:7*i));
    si=zeros(3,1);
    for k=1:m
        pro=and(h(k,:),cuv);
        si(k)=mod(sum(pro),2);
    end
    if sum(si)~=0
        for j=1:n
            prov=xor(si,h(:,j));
            if sum(prov)==0
                k1=j;
            end
        end
        biti(7*(i-1)+k1)=xor(1,biti(7*(i-1)+k1));    %
corectare!
    end
end    % se localizeaza eroarea si se corecteaza
b=zeros(1,nc);
for i=1:nc
    a1=[biti(14*(i-1)+1:14*i-7-3) biti(14*(i-1)+8:14*i-3)];
    b(i)=binvec2dec(a1);
end
disp(' ')
disp('Textul cu eroarea corectata:')
disp(' ')
disp(char(b))
disp(' ')
```

Tema 3: Coduri ciclice

Se propune spre executie si studiu scriptul Matlab următor:

```
clear
n=15;k=10; % lungimea cuvintelor de cod si lungimea cuvintelor
informationale
sn=int2str(n);sk=int2str(k);
s=strcat('Se determina toate polinoamele de grad
(',sn,'-',sk,')');
s=strcat(s,' generatoare ale unui cod cicilic (',sn,',',sk,')');
disp(' ')
disp(s)
pol=cyclpoly(n,k,'all')
pause
s=strcat('Se genereaza matricea de verificare si matricea
generatoare');
disp(s)
[H,G]=cyclgen(15,pol(2,:))
wt=gfweight(G);
swt=int2str(wt);
disp(strcat('Ponderea codului: ',swt))
pause
disp(' ')
disp('Un cuvant informational:')
info=dec2binvec(12,10)
disp('Cuvantul de cod asociat:')
c=mod(info*G,2)
sindrom=mod(H*c',2)
pause
c1(2:15)=c(1:14); % o permutare circulara a cuvantului de cod
c1(1)=c(15);
disp('Cuvantul de cod si o permutare circulara a lui')
[c' c1']'
sindrom=mod(H*c1',2)
pause
disp('Cuvintele de mai sus si suma lor')
c2=mod(c+c1,2);
[c' c1' c2']'
sindrom=mod(H*c2',2)
pause
disp('Se introduce o eroare pe bitul al cincilea')
cer=c;
cer(5)=xor(c(5),1);
[c' cer']'
sindrom=mod(H*cer',2)
pause
disp('Se introduc pana la patru erori pe pozitii aleatoare')
for i=1:10
    r=unidrnd(15,1,4);
    r=sort(r)
    cer=c;
    for j=1:4
        cer(r(j))=xor(c(r(j)),1);
    end
```

```
    [c' cer']'
    sindrom=mod(H*cer',2)
    if sindrom==zeros(n-k,1)
        disp('Incapacitatea codului de a detecta mai mult de
trei erori')
    end
    pause
end
```

Problema 64

Informatia generată de o sursă se poate reprezenta prin numere binare de lungime egală cu 7 biti. Se adaugă un bit suplimentar, un bit de paritate, prin care se obtine un cod de paritate constantă care include si cuvântul de cod nul.

a. Arătati că ponderea Hamming a acestui cod este 2.

Solutie: În evaluarea ponderii unui cod, cuvântul nul se omite dacă el apartine codului. Aici cuvântul nul, 00000000, este de o paritate care-l face să apartină codului. Se caută un cuvât de pondere 1. Asemenea cuvinte de 8 biti există dar nu apartin codului deoarece paritatea lor nu corespunde. În schimb, cuvinte de acceasi lungime, de pondere 2 există si apartin codului: prezenta a două (si numai două) unităti binare asigură paritatea necesară. Concluzia: ponderea codului din enunt este 2.

b. Arătati că acest cod este un cod liniar.

Solutie (1): Prin definitie, un cod este liniar dacă suma bit-cu-bit *modulo* 2 a două cuvinte de cod (oricare ar fi acelea) conduce todeauna la un cuvânt care apartine codului. Dacă în cele două cuvinte pozitiile unităților binare sunt toate diferite, ponderea cuvântului rezultat este suma ponderilor cuvintelor însumate. Suma a două numere pare este tot un număr par, asadar cuvântul rezultat apartine codului. Dacă unele unităti binare coincid ca pozitie în cuvintele însumate, ele se anihilează în perechi si din suma ponderilor (număr par) se scad 2, 4, 6 etc. totdeauna un număr par. Rezultatul este o pondere pară si cuvântul rezultat apartine codului.

Solutie (2): Ecuatia

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 0$$

sumă *modulo* 2 a bitilor componenti, este sistemul liniar vrificat de cuvintele codului. Concluzia: codul este liniar.

Problema 65

Se dă un cod rectangular 3×4 cu dublă paritate, se dă cuvântul de cod

$$\begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{matrix} \equiv 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1$$

a. Arătati că ponderea Hamming a codului este 4.

Solutie: Prin verificare directă, ponderea nu poate fi 1, 2 sau 3 deoarece conditiile de paritate pe linii si pe coloane n-ar fi îndeplinite. Cuvinte cu ponderea 4 există si ponderea codului este 4.

b. Scrieti un cuvânt de cod situat la distanta Hamming 4 fată de cuvântul dat.

Solutie: Prin inversarea unui singur bit în partea informatională se obtine inversarea altor trei biti de paritate. Se obtine astfel un cuvânt (din mai multe posibile) care apartine codului si este la distanta Hamming 4 fată de cuvântul propus în enunt. Exemplu:

$$\begin{array}{cccc} 1 & 0 & \underline{0} & \underline{1} \\ 0 & 1 & \underline{0} & 1 \\ 1 & 1 & \underline{0} & \underline{0} \end{array} \equiv 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0$$

c. Scrieti matricea de control a codului.

Solutie: Matricea cerută rezultă din scrierea conditiilor de paritate pe linii si pe coloane. Rezultă un sistem omogen de 7 ecuatii cu 12 necunoscute verificat numai de cuvintele codului. Coeficientii sistemului fac matricea de control. Atentie! Din cele 7 ecuatii numai 6 sunt independente, una din ele, cea de a saptea rezultă din celelalte.

d. Se receptionează cuvântul

1 0 1 0 1 1 0 1 1 1 1 1

El nu apartine codului deoarece contine o eroare. Corectati eroarea.

Solutie: Se scrie codul în forma rectangulară.

$$1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1 \equiv \begin{array}{cccc} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ \uparrow & & & \end{array} \leftarrow$$

Pe linia si coloana marcate cu săgeti paritatea este necorespunzătoare. Bitul eronat se află pe linia si pe coloana marcate. Inversarea lui rezolvă eroarea. Cuvântul corect(at) este

1 0 1 0 0 1 0 1 1 1 1 1

Problema 66. Coduri Golay

Detinând cunostinte exclusiv despre codul Hamming (7, 4), Marcel J.E.Golay a generalizat ideea lui Hamming la coduri perfecte corectoare de o eroare cu baza orice număr prim. După ce a terminat acest aspect, Golay a început să caute coduri perfecte corectoare de mai multe erori. Unul din codurile Golay este de interes special deorece a fost utilizat mai târziu pentru a genera pachete în spatiul 24-dimensional.

Se reaminteste că un cod corector de x erori trebuie să fie caracterizat de o distantă Hamming minimă de cel putin $2x + 1$. Pentru ca un cod să fie perfect, numărul de vârfuri ale unui n-cub unitar dintr-o sferă de rază x care-l cuprinde trebuie să fie o putere a lui r, unde r este *rădăcina codului*. În cazul binar

$$\sum_{i=0}^{x} C_n^i = 2^k$$

pentru un anumit întreg k. Aceasta este suma primilor $x + 1$ coeficienti/elemente din linia a n-a a triunghiului lui Pascal. Golay a găsit două asemenea numere

$$\sum_{i=0}^{2} C_{90}^{i} = 2^{12} \quad \text{si} \quad \sum_{i=0}^{3} C_{23}^{i} = 2^{11}$$

În cazul $n = 90$ Golay a arătat că nu există nici un cod perfect (90, 78) corector a două erori. Pentru $n = 23$ Golay a găsit un cod (23, 12) corector de 3 erori si a si dat o matrice pentru el

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Prin alipirea la aceasta a matricii unitate I_{11}, se obtine o matrice de verificare de aceeasi formă cu acelea pentru codurile corectoare de o eroare.

Pentru a construi codul, din digitii mesajului $y_1, \ldots, y_r$ se evaluează digitii de control (de paritate) $x_1, \ldots, x_k$ astfel încât

$$x_i + \sum_{j=1}^{12} a_{ij} y_j \equiv 0 \bmod 2 \,, \; i = 1, 2, \ldots, 11$$

sau

$$x_i \equiv \sum_{j=1}^{12} a_{ij} y_j \bmod 2 \,, \; i = 1, 2, \ldots, 11$$

Cuvintele de cod au forma $y_1, \ldots, y_r, x_1, \ldots, x_k$ cu digitii de paritate plasati în partea din urmă a cuvântului.

a. Stabiliti ponderea codului.

Solutie: Cu scriptul Matlab următor se generează toate cuvintele de cod nenule, în număr de $2^{12} - 1$ si se retine ponderea cea mai mică.

```
clear
% Se initializeaza matricea conform enuntului
a=[1 0 0 1 1 1 0 0 0 1 1 1;1 0 1 0 1 1 0 1 1 0 0 1
    1 0 1 1 0 1 1 0 1 0 1 0;1 0 1 1 1 0 1 1 0 1 0 0
    1 1 0 0 1 1 1 0 1 1 0 0;1 1 0 1 0 1 1 1 0 0 0 1
    1 1 0 1 1 0 0 1 1 0 1 0;1 1 1 0 0 1 0 1 0 1 1 0
    1 1 1 0 1 0 1 0 0 0 1 1;1 1 1 1 0 0 0 0 1 1 0 1
    0 1 1 1 1 1 1 1 1 1 1 1]
po=23;  % Se initilizeaza ponderea la valoarea maxima posibila
% Se genereaza toate cuvintele de cod nenule
% si se calcualeaza ponderea fiecaruia
```

```
for i=1:2^12-1
    ci=dec2bin(i,12);
    for k=1:12
        b(k)=str2num(ci(k));
    end
    c=(mod(a*b',2))';
    cc=[b c];
    po=min(po,sum(cc));
end
% Se retine si se afiseaza ponderea minima
po
```

Ponderea codului Golay este 7. De aici capacitatea codului de a corecta până la 3 erori.

b. Stabiliti codomeniul unui cuvânt de cod la alegere.

Solutie: Fie acel cuvânt de cod 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 1 0 1. Se retin ca apartinând codomeniului cuvintele de cod la distanta Hamming cel mult egală cu 3. Însumate, aceste cuvinte care diferă prin 1, 2 sau 3 biti sunt în număr de

$$C_{23}^1 + C_{23}^2 + C_{23}^3 = 23 + 253 + 1771 = 2047$$

la care se adaugă cuvântul însusi.

Codomeniul oricărui cuvânt de cod este alcătuit din 2^{11} cuvinte de 23 de biti.

Problema 67. Cod Hamming extins

Se defineste codul Hamming extins H_8 de lungime n = 8 ca o multime de secvente (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) de 8 valori binare care satisfac (*modulo* 2) următorul sistem de ecuatii liniare simultane:

$$x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 0 \qquad (1)$$
$$x_0 + x_1 + x_2 + x_3 = 0 \qquad (2)$$
$$x_0 + x_1 + x_4 + x_5 = 0 \qquad (3)$$
$$x_0 + x_2 + x_4 + x_6 = 0 \qquad (4)$$

S-a demonstrat că acest cod Hamming extins este capabil a corecta cel putin 1 eroare printr-o regulă de decodare explicită (bazată pe un tabel de sindromuri) capabilă a corecta orice eroare pe un singur bit.

a. Listati toate cuvintele de cod din H_8.

Solutie: Cuvintele codului sunt solutii ale sistemului de ecuatii omogen din enunt. Matricea sistemului este

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Prin operatii cu linii se obtine o formă sistematică:

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dacă se aleg ca biti de paritate x_0, x_2, x_3 si x_7, după ce x_7 devine x_1 si x_1 devine x_7, se obtine matricea

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

tot sistematică dar cu bitii de paritate separati de bitii informationali.
Cuvintele de cod se obtin numărând pe 4 biti de la 0 la 15 si multiplicând cuvintele obtinute cu matricea A.

b. Determinati o matrice generatoare pentru H_8.

Solutie: Matricea generatoare este matricea A în forma ultimă obtinută la punctul anterior.

c. Determinati o matrice de verificare pentru H_8.

Solutie: Aceeasi matrice A este si matrice de verificare.

d. Determinati dimensiunea si rata codului H_8.

Solutie: Cuvintele de cod au lungimea 8, rata codului este 8/4 = 2.

Problema 68.

Se ia în considerare din nou codul Hamming extins H_8 în versiunea definită în **Problema 67**.

a. Eliminati prima restrictie, ecuatia (1) si definiti un nou cod H_8^* alcătuit din secvente de 8 valori binare care satisfac (doar) restrictiile (2), (3), (4). Care este dimensiunea si care este rata acestui nou cod? Este (si) H_8^* capabil a corecta 1 eroare? Explicati matematic.

Solutie: Matricea sistemului verificat de cuvintele de cod este în această variantă

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

cu bitul x_7 liber de orice restrictie.
Dimensiunea codului este 8, rata este 8/4 = 2. Codul nu este capabil a face diferenta între corectitudinea si incorectitudinea bitului ultim, x_7.

b. Eliminati acum restrictia (1) si stergeti bitul x_7 pentru a obtine un cod nou H_7 alcătuit din secvente mai scurte, de 7 biti (operatia se numeste perforarea codului H_8). Care sunt dimensiunea si rata noului cod? Este (încă) acest cod capabil a corecta 1 eroare? Explicati matematic.

Solutie: Renuntarea la prima restrictie si la bitul x_7 transformă codul H_8^* în codul H_7. Dimensiunea noului cod este 7 si rata este 7/4 = 1,75. Da, codul este capabil a corecta o eroare. Codul devine un cod Hamming H_7.

Problema 69.

Se consideră din nou codul H_7 dezvoltat în problema precedentă. Arătati că orice secventă posibilă de 7 biti receptionată poate fi convertită într-un cuvânt de cod prin inversarea a cel mult unuia din bitii ei.
Solutie: Se calculează un sindrom care este un vector cu trei componente binare. Dacă sindromul este nul, nu trebuie inversat nici un bit. Un sindrom nenul – sunt sapte asemenea vectori sindrom – este criteriul de corectie pentru codul Hamming H_7 corector de o eroare: se inversează bitul de pe pozitia coloanei din matricea de verificare care coincide cu sindromul.

Problema 70.

Fie H_7 utilizat pentru un canal cu stergere BEC(p) – binary erasure channel – cu probabilitatea de stergere satisfăcând conditia $0 < p < 1$.
Arătai că H_7 este capabil a corecta toate tipurile de erori cu două sau mai putine stergeri.
Solutie: Codul H_7 are matricea de verificare în forma canonică (sistematică)

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

astfel că orice cuvânt de cod verifică ecuatiile de verificare a parităţii

$$x_0 = x_3 + x_5 + x_6$$
$$x_1 = x_3 + x_4 + x_6$$
$$x_2 = x_3 + x_4 + x_5$$

Se admite că stergerile apar în pozitiile x_i, x_j cu $i \neq j$. Indiferent de alegerea indicilor i si j, există cel putin o ecuatie în care apare x_i dar nu x_j sau apare x_j dar nu x_i. O astfel de ecuatie se foloseste pentru a obtine bitul din acea pozitie modificat prin stergere. Orice altă ecuatie din cele de verificare a parităţii poate fi folosită pentru restabilirea celuilalt bit sters. Algoritmul acesta simplu arată că H_7 este capabil a corecta toate cuvintele afectate de cel mult două stergeri.

Problema 71.

Se consideră următoarele matrici 3×6:

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Este vreuna dintre aceste matrici generatoare a unui cod liniar binar (6, 3)?
Sunt printre ele matrici care generează acelasi cod?
Solutie: Matricea A este în forma canonică. Pentru a aduce pe B în forma canonică se procedează pas cu pas la: adunarea liniei prime la a treia, adunarea liniei a doua la prima si, în final, adunarea liniei a treia la prima si la a doua (eliminarea Gauss):

$$B \sim \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Matricea aceasta coincide cu A.
Si pentru a aduce matricea C la forma canonică se aplică tot eliminarea Gauss: se adună prima linie la a treia, apoi se adună linia a treia la prima si la a doua:

$$C \sim \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Forma canonică a matricei C este diferită si de A si de B în forma lor canonică comună. Toate cele trei matrici au rangul 3, asadar toate generează coduri (6, 3). A si B genereză acelasi cod, C generează un cod diferit.

Problema 72.

Se consideră un cod rectangular C construit din H_7 astfel:

- Se aranjează 16 biti informationali într-o matrice 4×4
- Se face codarea H_7 pe fiecare linie, ceea ce conduce la o matrice 4×7
- Se face acelasi lucru pe fiecare coloană pentru a crea o matrice 7×7
- Codul C constă în toate matricile 7×7 obtinute în acest mod.

Determinati distanta minimă Hamming/ponderea pentru codul C.
Determinati multimea tuturor parametrilor τ_c si τ_d pentru care C este un cod corector/detector de (τ_c, τ_d) erori, cu τ_d numărul cel mai mare posibil pentru un τ_c ales.
Solutie: În scrierea matricială, codul arată astfel

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} \\ x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} & x_{20} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \\ x_{28} & x_{29} & x_{30} & x_{31} & x_{32} & x_{33} & x_{34} \\ x_{35} & x_{36} & x_{37} & x_{38} & x_{39} & x_{40} & x_{41} \\ x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} & x_{48} \end{bmatrix}$$

cu partea informațională plasată în submatricea 4×4 din coltul de sud-est. Conform enuntului, codul trebuie să satisfacă pe ultimele patru linii restrictiile

$$x_{21+i} = x_{24+i} + x_{26+i} + x_{27+i}$$
$$x_{22+i} = x_{24+i} + x_{25+i} + x_{27+i}$$
$$x_{23+i} = x_{24+i} + x_{25+i} + x_{26+i}$$

pentru i = 0, 7, 14, 21, adică 12 ecuatii.

Analog, pe coloane, trebuie să satisfacă restrictiile

$$x_{0+i} = x_{21+i} + x_{35+i} + x_{42+i}$$
$$x_{7+i} = x_{21+i} + x_{28+i} + x_{42+i}$$
$$x_{14+i} = x_{21+i} + x_{28+i} + x_{35+i}$$

pentru i = 0, 1, 2, 3, 4, 5, 6, adică alte 21 de ecuatii. Totalul restrictiilor: 33. Precizarea unui cuvânt informational conduce la determinarea unică a bitilor de protectie/control. Cuvântul informational nul conduce la cuvântul de cod nul. Dacă un singur bit informational este nenul, fie acela x_{24}, atunci alti biti din partea de control devin nenuli. În cazul x_{24} = 1 (în exclusivitate), mai sunt nenuli bitii x_{21}, x_{22}, x_{23}, x_3, x_{10}, x_{17} dar si x_0, x_1, x_2, x_7, x_8, x_9, x_{14}, x_{15}, x_{16}. Acesta este un cuvânt cu ponderea 16. Dar nu toate cuvintele cu un singur bit nenul în partea informațională au ponderea 16. Dacă bitul nenul este x_{25}, atunci cuvântul de cod are bitii x_{22}, x_{23}, x_4, x_{11}, x_{18} si x_1, x_2, x_8, x_9, x_{15}, x_{16} nenuli. Acesta este un cuvânt cu ponderea 12. Si încă, dacă x_{32} este bitul informational nenul, cuvântul de cod are nenuli bitii x_{29}, x_{30}, x_{11}, x_{18} si x_8, x_{15}, x_9, x_{17}. Ponderea acestui cuvânt este 9. Ce se poate afirma pe baza acestor explorări este că ponderea codului nu este mai mare decât 9.

Pe de altă parte, orice linie din matricea de mai sus, care are în partea informațională cel putin un bit nenul este un cuvânt din codul Hamming H_7 cu ponderea de cel putin 3, ca orice cuvânt nenul din H_7. Tot asa, orice coloană care contine o unitate binară este un cuvânt din H_7, de asemenea cu ponderea 3 sau mai mare. Din aceste două afirmatii rezultă că ponderea codului propus este cel putin 3×3 = 9.

Cazul particular evidentiat mai sus, cel cu x_{32} = 1 si restul bitilor informationali nuli aduce precizarea că ponderea codului este chiar 9.

Perechile (τ_c, τ_d) pot fi: (0, 8), (1, 6), (2, 4), (3, 1), (4, 0).

Problema 73.

Pentru fiecare cod de lungime n = 2 la 12, elaborati un tabel al codurilor bloc liniare sistematice de dimensiune k = 2 cu cea mai bună posibil distantă Hamming minimă d_{min}. Tabelul trebuie să includă o matrice generatoare pentru cod ca si valoarea d_{min}.

Solutie: Cel mai simplă cale de tratare (si suficient de generală) este a considera următoarea structură a matricilor generatoare:

$$G = [I \quad | \quad P] = \left[\begin{array}{cc|ccccccccc} 1 & 0 & 1 & \cdots & 1 & 1 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 1 & \cdots & 1 & 1 & \cdots & 1 \end{array}\right]$$

cu P care are în prima linie b unităti consecutive aliniate la stânga si în a doua linie d unităti consecutive aliniate la dreapta.

Calculele specifice conduc la solutiile următoare:

$$n = 2 \;\rightarrow\; d_{min} = 1, \; G = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right]$$

$$n = 3 \;\rightarrow\; d_{min} = 2, \; G = \left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right]$$

$$n = 4 \;\rightarrow\; d_{min} = 2, \; G = \left[\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right]$$

$$n = 5 \;\rightarrow\; d_{min} = 3, \; G = \left[\begin{array}{cc|ccc} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

$$n = 6 \;\rightarrow\; d_{min} = 4, \; G = \left[\begin{array}{cc|cccc} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$$

$$n = 7 \;\rightarrow\; d_{min} = 4, \; G = \left[\begin{array}{cc|ccccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{array}\right]$$

$$n = 8 \;\rightarrow\; d_{min} = 5, \; G = \left[\begin{array}{cc|cccccc} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$n = 9 \;\rightarrow\; d_{min} = 6, \; G = \left[\begin{array}{cc|ccccccc} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$n = 10 \;\rightarrow\; d_{min} = 6, \; G = \left[\begin{array}{cc|cccccccc} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$n = 11 \;\rightarrow\; d_{min} = 7, \; G = \left[\begin{array}{cc|ccccccccc} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$n = 12 \;\rightarrow\; d_{min} = 8, \; G = \left[\begin{array}{cc|cccccccccc} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

Pentru k = 2 si un n oarecare din cele recomandate, este usor de arătat, dar destul de migălos, că cea mai bună distantă Hamming minimă este

$$d_{min} = \left\lfloor \frac{2n}{3} \right\rfloor$$

adică partea întreagă a numărului $\frac{2n}{3}$. Calculele prezentate mai sus pentru n = 2, 3, …, 11, 12 sustin acest rezultat.

Problema 74.

Se consideră C un cod binar [6, 3] cu matricea generatoare

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Regula de codare $c = (a_1, a_2, a_3)G$ nu este sistematică. Stabiliti un operator matricial J care să aplice cuvintele de cod (printr-un decodor) pe bitii de informatie corespunzători (pentru utilizatorul ultim), adică

$$(a_1, a_2, a_3) = cJ$$

Solutie: În exemplul acesta simplu, problema se poate rezolva prin încercări, poate prin inspectare repetată! Codurile mai lungi, mai complexe cer o tratare mai sistematică. Tratarea aceasta urmează.

Se poate folosi un artificiu din algebra liniară. Fie A o matrice $l \times m$ cu forma canonică B si fie $M = [A \mid I]$ matricea $l \times (m + l)$ obtinută prin alipirea matricei unitate I de dimensiune $l \times l$. Eliminarea Gauss aplicată matricei M conduce la forma ei canonică $[B \mid Q]$. Submatricea Q are o proprietate interesantă: $QA = B$. De fapt, Q "memorează" pasii eliminării Gauss care schimbă pe A în B.

Pentru motive care vor fi evidente mai departe, se aplică artificiul schitat mai sus matricei $A = G^T$. Astfel

$$M = [G^T \mid I] = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \sim \ldots$$

$$\ldots \sim \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Etapele parcurse până la forma ultimă, cea canonică constau în operatii cu linii si sunt următoarele:

- se adună linia 1 la liniile 2 si 5
- se adună linia 2 la linia 6 si linia 3 la linia 5
- se adună linia 2 la linia 3
- se adună linia 3 la linia 4
- se adună linia 3 la linia 1.

Matricile interesante sunt

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{si} \quad Q = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Din $c = aG$, rezultă $c^T = G^T a^T$. Se stie totodată că $QG^T = B$, astfel că

$$Qc^T = QG^T a^T = Ba^T = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Zerourile din vectorul ultim se pot ignora, asa încât fie Λ matricea 3×6 obtinută din primele trei linii ale matricei Q. Atunci $\Lambda c^T = a^T$ si solutia problemei este matricea

$$J = \Lambda^T = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Problema 75.

Fie un cod C care admite următorul decodor optimal (de distantă minimă)

Sindromul	**Eroarea**		**Sindromul**	**Eroarea**
$[0000]^T$	0000000		$[0001]^T$	0000001
$[1000]^T$	0001000		$[1001]^T$	0010100
$[0100]^T$	0000100		$[0101]^T$	0011000
$[1100]^T$	0001100		$[1101]^T$	0010000
$[0010]^T$	0000010		$[0011]^T$	0000011
$[1010]^T$	0001010		$[1011]^T$	0100000
$[0110]^T$	0000110		$[0111]^T$	1000000
$[1110]^T$	0111000		$[1111]^T$	0010010

a. Determinati o matrice generatoare pentru C.
Solutie: Matricea de verificare este

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} P & | & I_{n-k} \end{bmatrix}$$

si se stabileste prin rezolvarea câtorva ecuatii simple. De pildă cuvântul de cod afectat de eroarea 0001000 filtrează coloana $[1\ 0\ 0\ 0]^T$ din matricea H. Similar, alti vectori de eroare cu un singur bit nenul filtrează coloane din H care sunt identice cu vectorii sindrom. Vectorii eroare cu mai multi biti nenuli sunt utilizati pentru verificare.

Acum, matricea generatoare a codului se scrie preluând submatricea P din partitia lui H de mai sus si completând-o deasupra cu matricea unitate I_k:

$$\begin{bmatrix} I_k \\ - \\ P \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

b. Utilizati decodorul pentru a decoda vectorul receptionat $\bar{r} = [1110111]$.

Solutie: Evaluarea produsului $H\bar{r}^T$ produce sindromul $[0110]^T$. Conform tabelului, vectorul eroare este $e = [0000110]$ si

$$r = \bar{r} + e = [1110001]$$

în aritmetica potrivită (bit-cu-bit *modulo* 2).

Problema 76.

Se consideră codul C bloc liniar [7, 3], cu matricea generatoare în formă sistematică următoare:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Acest cod este destinat transmiterii informatiei codate sistematic pe un BSC (Binary Symmetric Channel) cu probabilitatea încrucisată p. Utilizati matricea standard[2] pentru C, dată imediat mai jos, pentru a răspunde la întrebările privitoare la performantele decodării.

[2] **Matricea standard**. În teoria codării, matricea standard (sau matricea slepian) este o matrice cu dimensiunile $qn-k \times qk$ care listează toate elementele unui spatiu vectorial particular $F_q n$. Matricea standard se utilizează pentru a decoda codurile liniare, adică pentru stabilirea cuvântului de cod corespunzător unui vector-mesaj primit. În cazul de fată, dar si în general, matricea standard are în prima linie cuvintele codului, în prima coloană vectorii eroare lideri de comultimi/codomenii care se regăsesc pe aceeasi linie. Celelalte elemente se obtin prin însumarea bit-cu-bit modulo 2 a vectorilor eroare cu cuvintele de cod, pe linia, pe coloana corespunzătoare.

Matricea standard

0000000	1000111	0101011	0011101	1101100	1011010	0110110	1110001
0000001	1000110	0101010	0011100	1101101	1011011	0110111	1110000
0000010	1000101	0101001	0011111	1101110	1011000	0110100	1110011
0000100	1000011	0101111	0011001	1101000	1011110	0110010	1110101
0001000	1001111	0100011	0010101	1100100	1010010	0111110	1111001
0010000	1010111	0111011	0001101	1111100	1001010	0100110	1100001
0100000	1100111	0001011	0111101	1001100	1111010	0010110	1010001
1000000	0000111	1101011	1011101	0101100	0011010	1110110	0110001
0000011	1000100	0101000	0011110	1101111	1011001	0110101	1110010
0000110	1000001	0101101	0011011	1101010	1011100	0110000	1110111
0001100	1001011	0100111	0010001	1100000	1010110	0111010	1111101
0011000	1011111	0110011	0000101	1101100	1000010	0101110	1101001
0001010	1001101	0100001	0010111	1100110	1010000	0111100	1111011
0010100	1010011	0111111	0001001	1111000	1001110	0100010	1100101
0010010	1010101	0111001	0001111	1111110	1001000	0100100	1100011
0111000	1111111	0010011	0100101	1010100	1100010	0001110	1001001

a. Determinati $P_U(E)$ probabilitatea nedetectării erorii într-un cuvânt.

Solutie: $P_U(E) = P$(vectorul eroare este un cuvânt de cod nenul). Numărătorul de pondere pentru cod este $A(x) = 1 + 7x^4$. Sunt sapte cuvinte de cod nenule, toate cu ponderea Hamming egală cu 4. Astfel,

$$P_U(E) = 7p^4(1-p)^3$$

Acelasi răspuns se obtine si cu formula

$$P_U(E) = (1-p)^n \left[A\left(\frac{p}{1-p} \right) - 1 \right]$$

b. Determinati $P(E)$ probabilitatea ca decodorul optimal prin distantă minimă să producă un cuvânt de cod gresit (eroare de decodor).

Solutie: P(E) = P(vectorul eroare nu este lider de codomeniu). Se tabelează distributia după pondere a liderilor de codomeniu:

Ponderea Hamming	0	1	2	3
Numărul de lideri de codomeniu	1	7	7	1

Atunci

$$P(E) = 1 - P(\text{vectorul de eroare este lider de codomeniu}) =$$
$$= 1 - (1-p)^7 - 7p(1-p)^6 - 7p^2(1-p)^5 - p^3(1-p)^4$$

c. Determinati probabilitatea ca decodorul optimal de distantă minimă să decodeze incorect toti cei trei biti de informatie.

Solutie: Se admite că se transmite c_{000} = (0000000). (Pentru alt cuvânt de cod retinut, analiza este similară). Toti cei trei biti de informatie vor fi decodati eronat atunci si numai atunci când decodorul produce cuvântul de cod c_{111} = (1110001). Astfel,

$$P(\text{toti cei trei biti de informatie sunt eronati}) =$$
$$= P(\text{vectorul eroare este în coloana care are în cap pe } c_{111})$$

Tabelarea distributiei după ponderi în acea coloană conduce la

Ponderea Hamming	0	1	2	3	4	5	6	7
Numărul de lideri de codomeniu				5	5	3	3	

Asadar,

$$P(\text{toti cei trei biti de informatie sunt eronati}) =$$
$$= 5p^3(1-p)^4 + 5p^4(1-p)^3 + 3p^5(1-p)^2 + 3p^6(1-p)$$

d. Determinati $P(F)$ probabilitatea ca un decodor de distantă $t = 1$ limitată să esueze si să nu producă vreun răspuns (cădere de decodor).

Solutie: $P(F) = P(\text{vectorul eroare este în liniile 9 – 16})$. Se tabelează distributia după ponderi a cuvintelor din aceste linii.

Ponderi	Număr de aparitii
2	21
3	7
4	28
5	0
6	7
7	1

Cu aceste observatii

$$P(F) = \sum_{i=2}^{7} N_i p^i (1-p)^{7-i} =$$
$$= 21p^2(1-p)^5 + 7p^3(1-p)^4 + 28p^4(1-p)^3 + 7p^6(1-p) + p^7$$

Acest rezultat corespunde formulei

$$P(F) = 1 - \sum_{i=0}^{t} C_n^i p^i (1-p)^{n-i} - P(E)$$

Problema 77.

Examinarea matricei standard din problema precedentă arată că s-au făcut unele alegeri arbitrare pentru cuvintele lider de comultimi/codomenii (cosets) (pattern-uri de eroare "corectabilă") în linia 9 si următoarele. (Posibilitătile diferite corespund diferentelor în ordonarea elementelor slab-ponderate din lista de bază (scratch) de la care pornind s-a construit matricea standard.)

a. În linia ultimă, se presupune că 1001001 a fost ales în locul lui 0111000 ca lider corect. Care va fi ultima linie a noii matrici standard? Calculati $P(E)$ pentru noul decodor cu noua matrice standard.

Solutie: Linia ultimă din tabel care era

0111000 | 1111111 0010011 0100101 1010100 1100010 0001110 1001001

devine acum

1001001	0001110	1100010	1010100	0100101	0010011	1111111	0111000

Probabilitatea $P(E)$ este aceeasi deoarce liderul ce codomeniu are aceeasi pondere ca mai înainte si linia este doar reordonată. În fapt, atât $P_U(E)$ cât si $P(E)$ rămân neschimbate.

b. Repetati punctul a. dar cu utilizarea lui 1111111 în loc de 0111000 ca lider de codomeniu. (De observat că această alegere nu conduce la o matrice standard! De ce?)

Solutie: În acest caz, noua linie ultimă este

1111111	0111000	1010100	1100010	0010011	0100101	1001001	0001110

si atunci

$$P(E) = 1 - P(\text{vectorul eroare este lider de codomeniu}) =$$
$$= 1 - (1-p)^7 - 7p(1-p)^6 - 7p^2(1-p)^5 - p^7$$

ceea ce este mai mult decât rata erorii pentru decodorul optimal (prin ditanta minimă).

Problema 78.

Un sistem ce comunicatie care utilizează coduri bloc operează continuu. Ocazional, echipamentul de transmitere greseste fără stirea receptorului. În asemenea împrejurări, canalul este în esentă un BSC cu $p = 1/2$. Pentru codurile următoare, care este probabilitatea ca un decodor de distantă t limitată să decodeze incorect un sir de biti aleator receptionat ca un cuvânt de cod valid?

a. Codul Hamming [15, 11] cu $t = 1$

Solutie: Probabilitatea de acceptare falsă este egală cu probabilitatea ca vectorul aleator primit să se situeze în sfera de corectabilitate din jurul unui anumit cuvânt de cod. În general, această probabilitate poate fi calculată ca raportul numărului total de vectori din toate aceste sfere si numărul total de vectori posibili. Astfel,

$$P(\text{acceptare falsă}) = [M \cdot V(n, t)/2^n] = V(n, t)/2^r$$

Codurile Hamming sunt perfecte si sunt si perfecte umpleri cu sfere a spatiului cuvintelor de lungime n. Asadar, $P(\text{acceptare falsă}) = 1$.

b. Codul liniar binar [23, 12] cu $t = 3$

Solutie: Pentru un cod binar liniar [23, 12] cu $t = 3$, se execută calculul

$$V(23, 3) = C_{23}^0 + C_{23}^1 + C_{23}^2 + C_{23}^3 = 1 + 23 + 253 + 1771 = 2048$$

Asadar,

$$P(\text{acceptare falsă}) = V(23, 3)/2^{32-12} = 2048/2048 = 1$$

La acest punct este vorba de codul Golay.

Si codul Golay el face o umplere perfectă cu sfere a spatiului cuvintelor binare de lungime 23.

c. Codul liniar binar [63, 36] cu $t = 6$.

Solutie: În acest caz, se calculează

$$V(63, 6) = C_{63}^0 + C_{63}^1 + C_{63}^2 + C_{63}^3 + C_{63}^4 + C_{63}^5 + C_{63}^6 =$$
$$= 1 + 63 + 1953 + 39711 + 595665 + 7028847 + 67945521 = 75611761$$

astfel că

$$P(\text{acceptare falsă}) = V(63, 6)/2^{63-36} = 0{,}56335153$$

Problema 79.

Fie C un cod ciclic de lungime 15 generat cu polinomul generator $g(x) = x^5 + x^4 + x^2 + 1$.

a. Verificati că $g(x) \mid x^{15} + 1$.

Solutie: Se verifică fără dificultate că

$$(x^5 + x^4 + x^2 + 1)(x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1) = x^{15} + 1$$

b. Determinati numărul de cuvinte de cod din C.

Solutie: Se stie că $r = \text{grd}[g(x)] = 5$ astfel că se codează cuvinte informationale de lungime $k = n - r = 15 - 5 = 10$. Asadar codul C este alcătuit din $2^{10} = 1024$ cuvinte.

c. Determinati matricea generatoare sistematică si matricea de verificare a parității pentru C.

Solutie: Matricea generatoare G_1 în forma ei nesistematică se obtine prin completarea liniei i din G_1 ($i = 1, 2, \ldots, 10$) cu coeficientii polinomului $1 + x^2 + x^4 + x^5$ multiplicat cu x^{i-1}. Astfel se obtine

$$G_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Întrucâtva analog se obtine si matricea de verificare a parității H_1 asociată cu G_1, prin completarea liniilor lui H_1 cu coeficientii polinomului $x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1$ aliniati la dreapta pe prima linie si deplasati cu câte o pozitie spre stânga în liniile următoare.

$$H_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Cu o migală relativă se poate verifica egalitatea

$$G_1 \times H_1^T = 0$$

echivalentă în fond cu descompunerea

$$(x^5 + x^4 + x^2 + 1)(x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1) = x^{15} + 1$$

Matricea generatoare în forma sistematică/canonică G se poate obtine fie prin operatii cu linii pe matricea G_1, fie direct prin completarea liniei i din G (i = 1, 2, …, 10) cu coeficientii polinomului $c_i(x) = x^{r+i-1} + \text{rest}\{x^{r+i-1}/g(x)\}$. (Se propune ca exercitiu verificarea identitătii rezultatelor obtinute pe cele două căi.).

Într-un mod sau altul se ajunge la

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Pentru a obtine matricea de verificare a parităţii pentru codul sistematic se aplică regula obisnuită de trecere de la G la H: se extrage din G submatricea de la stânga submatricei unitate I_k, se transpune si i se alătură la stânga matricea unitate I_r.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

d. Executati codarea sistematică a polinomului mesaj $m(x) = x^9 + x^4 + x^2 + 1$ utilizând reprezentarea polinomială.

Solutie: Codarea sistematică[3] conduce la

$$c(x) = x^5(x^9 + x^4 + x^2 + 1) + \text{rest}\{x^5(x^9 + x^4 + x^2 + 1)/g(x)\} =$$
$$= x^{14} + x^9 + x^7 + x^5 + x^4 + x^3 + x^2 + 1$$

Cuvântul de cod obtinut: 101111010100001.

---

[3] Alternativ, pentru a crea un cod *sistematic*, se folseste adesea metoda următoare:
Fiind dat un cuvânt de date $d(x)$ de lungime $n - m$, se multiplică mai întâi $d(x)$ cu x^m, ceea ce are ca efect deplasarea lui $d(x)$ cu m pozitii spre stânga. În general, $x^m.d(x)$ nu este divizibil cu $g(x)$, deci nu este un cuvânt de cod. Există totusi un cuvânt de cod unic care se poate obtine prin ajustarea a m simboluri cele mai din dreapta ale polinomului $x^m.d(x)$. Pentru această operatie se calculează restul împărtirii lui $x^m.d(x)$ prin $g(x)$, $x^m.d(x) = g(x).q(x) + r(x)$, unde $r(x)$ este de grad mai mic decât m.
Cuvântul de cod care corespunde cuvântului de date $d(x)$ este atunci $p(x) = x^m.d(x) + r(x)$

e. Calculati sindromul pentru polinomul mesaj receptionat $r(x) = x^{10}$ utilizând reprezentarea polinomială.

Solutie: Conform matricei de verificare a paritătii în forma sistematică, sindromul se calculează ca

$$s(x) = \text{rest}\{x^{10}/g(x)\}$$

Deoarece

$$x^{10} = (x^5 + x^4 + x^3 + x + 1)g(x) + (x^2 + x + 1),$$

sindromul în formă polinomială este

$$s(x) = 1 + x + x^2.$$

Problema 80.

Fie c_1 si c_2 cuvinte de cod din codul liniar C.

a. Exprimati ponderea Hamming a cuvântului $c = c_1 + c_2$ în raport cu ponderile Hamming ale cuvintelor c_1, c_2 si $c_1 \wedge c_2$ cu $\wedge$ operatorul AND (ŞI) bit-cu-bit.

Solutie: Se observă că între ponderile cuvintelor există relatia

$$\text{wt}(c) = \text{wt}(c_1) + \text{wt}(c_2) - 2\text{wt}(c_1 \wedge c_2)$$

b. Arătati că suma a două cuvinte de cod de pondere pară din C este de pondere pară.

Solutie: În operatia de adunare a două cuvinte de cod nu apar biti unitari noi, iar cei care dispar dispar în perechi.

c. Arătati că suma a două cuvinte de cod de pondere impară din C este de pondere pară.

Solutie: Aceeasi observatie de la punctul anterior, cu adaosul că suma a două numere impare are ca rezultat un număr par.

d. Arătati că suma a două cuvinte de cod de pondere pară si impară din C este de pondere impară.

Solutie: Ca si la punctele anterioare, suma unui număr par cu unul impar este un număr impar. Ceilalti termeni din relatia de la primul punct nu schimbă paritatea.

Problema 81.

Într-un cod liniar, multimea cuvintelor de cod de pondere Hamming pară formează un cod mai restrâns care este tot liniar (problema anterioară). El este numit *subcodul de pondere pară*.

Fie C un cod ciclic de lungime n, de dimensiune k, care are ca generator polinomul $g(x)$. Se admite că $x + 1$ nu este factor al polinomului $g(x)$. Arătati că un cod ciclic generat de $(x + 1)g(x)$ este subcod al lui C, de pondere pară. (*Indicatie*: $c(x)$ este de pondere Hamming pară dacă si numai dacă $c(1) = 0 \bmod 2$.)

Solutie: Fie $g_1(x) = (x + 1)g(x)$ si C_1 codul ciclic pe care acest polinom îl generează. Se observă că $x + 1$ si $g(x)$ sunt factori ai polinomului $x^n + 1$ fără să aibă vreun factor comun neconstant. Produsul $(x + 1)g(x)$ este de asemenea

factor al polinomului $x^n + 1$ asa încât codul C_1 este un cod ciclic. Urmează acum a demonstra că:

($\Rightarrow$) **orice cuvânt de cod din C_1 este cuvât de cod în C si este de pondere pară**.

Orice cuvânt de cod din C_1 este cuvânt de cod în C deoarece orice multiplu al lui $g_1(x)$ este în acelasi timp multiplu al lui $g(x)$. Se retine acum un cuvânt de cod $c_1(x) = m(x)g_1(x)$. Ținând seamă de *Indicatia* atasată la enuntul problemei, se constată că $\text{wt}[c_1(x)] = m(1)g_1(1) = (1 + 1)m(1)g(1) = 0 \bmod 2$.

($\Leftarrow$) **orice cuvânt de cod din C de pondere pară este cuvânt de cod din C_1.**

Afirmatie: orice polinom de pondere pară este multiplu al lui $x + 1$. Pentru demonstratie se presupune că $p(x)$ este de pondere pară. Prin împărtirea lui $p(x)$ la $x + 1$ se obtine câtul $q(x)$ si restul r care satisfac teorema împărtirii întregi

$$p(x) = (x + 1)q(x) + r$$

cu r un polinom de gradul zero (o constantă).

Atunci,

$$0 = \text{wt}[p(x)] = p(1) = (1 + 1)q(1) + r = r$$

Asadar, $p(x) = (x + 1)q(x)$, adică este un multiplu de $(x + 1)$.

Acum se presupune că $c(x) = m(x)g(x)$ este un cuvânt de cod de paritate pară din C. Rezultă $c(1) = m(1)g(1) = 0$. Deoarece $g(x)$ nu este divizibil cu $x + 1$, are loc $g(1) = 1$. Astfel, $m(1) = 0$ si $x + 1$ este divizor al lui $m(x)$. Fie $m(x) = m_1(\text{x})(x + 1)$. Atunci $c(x) = m_1(x)(x + 1)g(x) = m_1(x)g_1(x)$ este cuvânt de cod din C_1.

Problema 82.

Două coduri se numesc echivalente dacă se pot permuta coordonatele vectoriale într-un mod în care cuvintele de cod ale unui cod se transformă în cuvintele de cod ale celuilalt cod. De pildă codurile generate de matricile

$$G_1 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix} \text{ si } G_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

sunt echivalente: primii doi biti sunt schimbati între ei. Performanta codurilor echivalente este aceeasi pe canalele fără memorie de genul BSC. Explicati cum codul ciclic generat de $g(x) = x^3 + x + 1$ este echivalent cu un cod H_7 definit în **Problema 70**, prin stabilirea permutării care schimbă un cod în altul.

Solutie: Din problema anterioară mentionată se extrage faptul că versiunea canonică a martricei de verificare a paritătii pentru codul Hamming H_7 este

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Matricea generatoare corespunzătoare este

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Această matrice este diferită de matricea generatoare canonică a versiunii ciclice a codului H_7 produsă de polinomul generator $g(x) = x^3 + x + 1$

$$G_{ciclic} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Prin inspectie, se poate vedea că primul bit de informatie (coloana a patra) din versiunea necicilică este bitul de informatie al treilea din versiunea ciclică (coloana a sasea); bitii de informatie al doilea si al doilea sunt aceeasi; bitul de informatie al treilea în versiunea neciclică este bitul de informatie ultim din versiunea ciclică si ultimul bit de informatie din versiunea neciclică este primul bit de informatie din varianta ciclică. Cu alte cuvinte prin schimbarea între ele a acestor coloane si prin rearanjarea liniilor pentru a restabili forma canonică se poate trece de la codul neciclic la cel ciclic.

Problema 83.

Pentru a genera un cod perfect Golay G_{23} se pot utiliza două polinoame generatoare, $g_1(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$ si $g_2(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$.

a. Arătati că $g_1(x)$ si $g_2(x)$ sunt polinoame inverse unul altuia.

Solutie: Polinomul cu ordinea coeficientilor inversată fată de cea a polinomului $g_2(x)$ este

$$x^{11} \cdot g_2\left(\frac{1}{x}\right) = x^{11} \cdot \left(\frac{1}{x^{11}} + \frac{1}{x^9} + \frac{1}{x^7} + \frac{1}{x^6} + \frac{1}{x^5} + \frac{1}{x} + 1\right) =$$
$$= 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$$

adică exact $g_1(x)$.

b. În ce relatie sunt cele două versiuni de G_{23}? Sunt codurile generate de $g_1(x)$ si $g_2(x)$ identice? Sunt echivalente? De ce?

Solutie: Cele două versiuni NU sunt identice deoarece polinomul generator al unui cod ciclic nu este TOTDEAUNA unic. Dar cele două versiuni ale codului G_{23} diferă numai prin ordinea de prezentare a bitilor: cuvintele de cod dintr-o versiune sunt cele ale celeilalte versiuni scrise în ordine inversată. Fie de pildă $c(x) = m(x)g_1(x)$ un cuvânt al codului generat de $g_1(x)$. Scris în ordine inversă el devine

$$c_{back}(x) = x^{n-1}\, c(1/x) = x^{n-1}\, m(1/x)\, g_1(1/x) = [x^{k-1}\, m(1/x)][x^r\, g_1(1/x)] =$$
$$= m_{back}(x)\, g_2(x)$$

cu $n = 23$, $k = 12$, $r = 11$ si cu $m_{back}(x)$ polinomul informational scris în ordinea inversă a coeficientilor. Schimbarea este o simplă permutare a coordonatelor astfel că cele două versiuni ale codului Golay sunt echivalente. (De retinut că "scris invers" este întrucâtva diferit de "inversat" deoarece avem o dimensiune fixată a blocului independentă de gradul polinomului.)

Problema 84.

O succesiune (explozivă) de erori (*error burst*) de lungime l este un pattern de erori $\bar{e} = (e_0, e_1, \ldots, e_{n-1})$ în care pentru o parte din indicii $0 \le i < n$ erorile sunt legate de pozitiile e_i, $e_{(i+1) \bmod n}$, …, $e_{(i+l-1) \bmod n}$ cu $e_i \neq 0$ si $e_{(i+l-1) \bmod n} \neq 0$, cu bitii intermediari eronati sau nu, cu ceilati biti, dinaintea primului si de după ultimul din secventa eronată, corecti. De pildă, pattern-urile (0010100) si (1100001) sunt ambele succesiuni (explozive) de lungime 3.
Fie $g(x)$ polinomul generator pentru un cod ciclic $C[n, k]$.

a. Arătati că C poate detecta toate erorile în succesiune (explozivă) de lungime $l \le n - k$.

Solutie: Fie $e(x)$ o eroare în succesiune de lungime $l \le r$, nedetectabilă. Deoarece e(x) este nedetectabilă, ea trebuie să fie un cuvânt de cod. Toate permutările circulare ale lui $e(x)$ sunt si ele cuvinte de cod si sunt totodată erori în succesiune de lungime l. Deoarece lungimea secventei (eronate) este l, există o permutare circulară $e^{(i)}(x)$ de gradul $l - 1$. Acel $e^{(i)}(x)$ este un cuvânt de cod monic de gradul $l - 1 < r$. Dar polinomul generator $g(x)$ este polinomul monic de gradul cel mai mic al codului si are gradul r. Contradictia apărută spune că un asemenea $e(x)$ nu există.

b. Arătati că nu toate erorile în succesiune (explozivă) de lungime $l = n - k + 1$ pot fi detectate.

Solutie: Polinomul $e(x)$ stabilit ca la punctul anterior este exact $g(x)$ si este asociat unei erori în succesiune de lungime $r + 1$. Eroarea este nedetectabilă deoarece $g(x)$ este un cuvânt de cod.

Problema 85.

Demonstrati următoarele fapte interesante relativ la polinoamele binare:

a. Dacă $p(x)$ este un polinom binar, atunci $(p(x))^2 = p(x^2)$.

Solutie: Fie $p(x) = a_0 + a_1x + \ldots + a_nx^n$. Urmează că $[p(x)]^2 = b_0 + b_1x + \ldots + b_{2n}x^{2n}$ cu coeficientii b_k obtinuti prin convolutia coeficientilor a ai polinomului $p(x)$ dată de

$$b_k = \sum_{i=0}^{k} a_i a_{k-i} \bmod 2$$

Mai detaliat, această sumă se poate scrie

$$b_k = a_0a_k + a_1a_{k-1} + \ldots + a_{k-1}a_1 + a_ka_0$$

Din simetria acestei expresii rezultă că prin însumare termenii extremi se anulează, la fel termenii extremi următori s.a.m.d. Dacă numărul de perechi este perfect, fără vreun termen neînperechiat, coeficientul b_k este nul. Asa se întâmplă când k este impar. Dacă k este par, de pildă $k = 2l$, atunci termenul central a_la_l nu se anulează, si $b_k = a_l^2 = a_l$. Astfel,

$$[p(x)]^2 = a_0 + a_1x^2 + \ldots + a_{n-1}x^{2n-2} + a_nx^{2n} = p(x^2).$$

b. Dacă $\alpha \in GF(2^r)$ este o rădăcină a unui polinom binar, atunci tot rădăcini sunt si α^2, α^4, α^8 etc.

Solutie: Dacă $p(\alpha) = 0$ atunci conform punctului anterior $p(\alpha^2) = [p(\alpha)]^2 = 0$ astfel încât α^2 este de asemenea o rădăcină. Iterând aceeasi idee, dacă α^2 este rădăcină, atunci $(\alpha^2)^2 = \alpha^4$ este rădăcină, la fel $(\alpha^4)^2 = \alpha^8$ este rădăcină s.a.m.d.

Problema 86.

Construiti un câmp Galois $GF(2^4)$ folosind polinoamele ireductibile date mai jos. Pentru ambele constructii determinati polinoamele minimale pentru fiecare element al câmpului respectiv.
Indicatie: Urmăriti cele două multimi de polinoame din cele două tabele deoarece ele sunt aceleasi în $GF(2^4)$.
a. $g_1(x) = x^4 + x + 1$
Solutie: Fie α o rădăcină a lui $g_1(x) = x^4 + x + 1$. Deoarece polinoml $g_1(x) = x^4 + x + 1$ este primitiv, puterile lui α dau toate elemente nenule din $GF(2^4)$. Polinoamele minimale pot fi aflate printr-o tratare sistematică prin încercări sau prin examinarea codomeniilor ciclotomice *modulo* 15. De exemplu, codomeniul ciclotomic al lui 3 mod 15 este $C(3) = \{3, 6, 12, 9\}$. Astfel, polinomul minimal al lui α^3 (ca si al conjugatelor sale α^6, α^9 si α^{12}) este

$$M_3(x) = (x + \alpha^3)(x + \alpha^6)(x + \alpha^9)(x + \alpha^{12}) = x^4 + x^3 + x^2 + x + 1$$

Similar, codomeniul ciclotomic al lui 5 mod 15 este C(5) = {5, 10} asa încât polinomul minimal pentru a5 si a10 este

$$M_5(x) = (x + \alpha^5)(x + \alpha^{10}) = x^2 + x + 1$$

Tabelul complet al elementelor câpului si al polinoamelor lor minimale este următorul:

Element	Polinom minimal	Element	Polinom minimal
0	x	$\alpha^7 = 1 + \alpha + \alpha^3$	$x^4 + x^3 + 1$
1	$x + 1$	$\alpha^8 = 1 + \alpha^2$	$x^4 + x + 1$
α	$x^4 + x + 1$	$\alpha^9 = \alpha + \alpha^3$	$x^4 + x^3 + x^2 + x + 1$
α^2	$x^4 + x + 1$	$\alpha^{10} = 1 + \alpha + \alpha^2$	$x^2 + x + 1$
α^3	$x^4 + x^3 + x^2 + x + 1$	$\alpha^{11} = \alpha + \alpha^2 + \alpha^3$	$x^4 + x^3 + 1$
$\alpha^4 = 1 + \alpha$	$x^4 + x + 1$	$\alpha^{12} = 1 + \alpha + \alpha^2 + \alpha^3$	$x^4 + x^3 + x^2 + x + 1$
$\alpha^5 = \alpha + \alpha^2$	$x^2 + x + 1$	$\alpha^{13} = 1 + \alpha^2 + \alpha^3$	$x^4 + x^3 + 1$
$\alpha^6 = \alpha^2 + \alpha^3$	$x^4 + x^3 + x^2 + x + 1$	$\alpha^{14} = 1 + \alpha^3$	$x^4 + x^3 + 1$

b. $g_2(x) = x^4 + x^3 + x^2 + x + 1$
Solutie: Fie β o rădăcină a polinomului neprimitiv $g_2(x) = x^4 + x^3 + x^2 + x + 1$ asa încât $\beta^4 = 1 + \beta + \beta^2 + \beta^3$. Puterile lui β nu produc toate elementele nenule ale câmpului $GF(2^4)$. Este necesar a explicita toate combinatiile liniare care lipsesc. Deoarece în această reprezentare nu există elemente primitive convenabile, pentru a stabili polinoamele minimale nu este posibil recursul la codomeniile ciclotomice modulo 15 cum s-a procedat la punctul anterior. De data aceasta conjugatele trebuie calculat direct. De pildă, conjugatele lui $\gamma = \beta + \beta^2$ sunt determinate astfel:

$$\gamma$$
$$\gamma^2 = (\beta + \beta^2)^2 = \beta^2 + \beta^4 = 1 + \beta + \beta^3$$
$$\gamma^4 = (\beta + \beta^2)^4 = \beta^4 + \beta^8 = 1 + \beta + \beta^2$$
$$\gamma^8 = (\beta + \beta^2)^8 = \beta^8 + \beta^{16} = \beta + \beta^3$$

Polinomul minimal pentru aceste elemente este

$$M_\gamma(x) = (x + \gamma)(x + \gamma^2)(x + \gamma^4)(x + \gamma^8) = x^4 + x + 1$$

Pe calea aceasta se găsesc elementele din tabelul următor si polinoamele lor minimale în reprezentarea β:

Element	Polinom minimal	Element	Polinom minimal
0	x	$1 + \beta^3$	$x^4 + x^3 + 1$
1	$x + 1$	$\beta + \beta^2$	$x^4 + x + 1$
β	$x^4 + x^3 + x^2 + x + 1$	$\beta + \beta^3$	$x^4 + x + 1$
β^2	$x^4 + x^3 + x^2 + x + 1$	$\beta^2 + \beta^3$	$x^2 + x + 1$
β^3	$x^4 + x^3 + x^2 + x + 1$	$1 + \beta + \beta^2$	$x^4 + x + 1$
$\beta^4 = 1 + \beta + \beta^2 + \beta^3$	$x^4 + x^3 + x^2 + x + 1$	$1 + \beta + \beta^3$	$x^4 + x + 1$
$1 + \beta$	$x^4 + x^3 + 1$	$1 + \beta^2 + \beta^3$	$x^2 + x + 1$
$1 + \beta^2$	$x^4 + x^3 + 1$	$\beta + \beta^2 + \beta^3$	$x^4 + x^3 + 1$

De observat că se obtin aceleasi rădăcini la acelasi set de polinoame. Aceasta din cauză că există numai un câmp $GF(2^4)$, indiferent cum este construit.

Problema 87.

Fie câmpul Galois $GF(2^4)$ construit prin adăugarea la câmpul simplu $F = \{0, 1\}$ a unei rădăcini α a polinomului primitiv $g(x) = x^4 + x^3 + 1$. Exprimati elementele γ de mai jos în functie de elementele de bază $\{1, \alpha, \alpha^2, \alpha^3\}$.
Indicatie: Se foloseste în calcule tabelul următor care contine elementele câmpului finit generat de primitivul $g(x)$.

0	$\alpha^7 = 1 + \alpha + \alpha^2$
1	$\alpha^8 = \alpha + \alpha^2 + \alpha^3$
α	$\alpha^9 = 1 + \alpha^2$
α^2	$\alpha^{10} = \alpha + \alpha^3$
α^3	$\alpha^{11} = 1 + \alpha^2 + \alpha^3$
$\alpha^4 = 1 + \alpha^3$	$\alpha^{12} = 1 + \alpha$
$\alpha^5 = 1 + \alpha + \alpha^3$	$\alpha^{13} = \alpha + \alpha^2$
$\alpha^6 = 1 + \alpha + \alpha^2 + \alpha^3$	$\alpha^{14} = \alpha^2 + \alpha^3$

a. $\gamma = 1 + \alpha^{-1} + \alpha^{-2} + \alpha^{-3}$
Solutie:

$$\gamma = 1 + \alpha^{-1} + \alpha^{-2} + \alpha^{-3} = 1 + \alpha^{14} + \alpha^{13} + \alpha^{12} = \alpha^3.$$

b. $\gamma = (\alpha^5(1 - \alpha^{-11}))/(1 + \alpha^{-2})$
Solutie:

$\gamma = (\alpha^5(1 - \alpha^{-11}))/(1 + \alpha^{-2}) = \alpha^5(1 + \alpha^4)/(1 + \alpha^{13}) = \alpha^5\alpha^3/\alpha^7 = \alpha$.

c. $\gamma = \sum_{\beta \in GF(2^4)} \beta$

Solutie: Este posibil ca punctele c. si d. să fie rezolvate prin calcul direct, ceea ce este un bun exercitiu în aritmetica câmpurilor finite. Există însă si o tratare mai rafinată. Fie $\{\beta_1, \beta_2, \ldots, \beta_{15}\}$ o listă completă a elementelor nenule din câmpul $GF(2^4)$. Prin dezvoltarea produsului tuturor factorilor de forma $(x + \beta_i)$, $i = 1, 2, \ldots, 15$, se obtine:

$(x + \beta_1)(x + \beta_2)\ldots(x + \beta_{15}) = x^{15} + (\beta_1 + \beta_2 + \ldots + \beta_{15})x^{14} + \ldots + (\beta_1\beta_2 \ldots \beta_{15})$.

Una din proprietătile câmpurilor Galois este aceea că un produs de acest gen este egal cu $x^{2^r-1} + 1$. Astfel,

$$\sum_{\beta \in GF(2^4)} \beta = 0$$

si, pentru punctul d.,

$$\prod_{\beta \in GF(2^4)-\{0\}} \beta = 1 .$$

d. $\gamma = \prod_{\beta \in GF(2^4)-\{0\}} \beta$.

Problema 88.

Se consideră o sursă de informatie binară pentru care iesirea este codată în blocuri de intrare de 52 de biti prin codul Reed-Soomon în sens restrâns de lungime 15 si distantă de proiectare 3. Utilizând regula de codare nesistematică $c(x) = m(x)g(x)$, determinati cuvântul de cod transmis pentru secventa informationalã binară

0001 0000 0000 0000 0000 0000 0000 0000 0000 1001 1110 0000 0101

Exprimati cuvântul de cod la iesire în notatia de la câmpuri finite (adică folosind elementele câmpului finit ales, cu α o primitivă), dar si ca o secventă binară. Fiti expliciti în ceea ce priveste ordinea bitilor si a simbolurilor cuvântului de cod alese!

Solutie: Pentru a evita orice ambiguitate, trebuie specificat modul cum este citită secventa de intare (de codat) ca un polinom din $GF(16)[x]$. Fiecare grupă de patru biti consecutivi trebuie interpretată ca o reprezentare în coordonate de patru biti a unui element al unui element din câmpul Galois în raport cu baza $\{1, \alpha, \alpha^2, \alpha^3\}$, în care α este o rădăcină a polinomului primitv $p(x) = x^4 + x + 1$. În fiecare simbol de patru biti, coeficientul binar al elementului din bază α^3 va fi numit MSB (Most Significant Bit); coeficientul binar pentru elementul din bază 1 va fi numit LSB (Least Significant Bit). Se admite că simbolurile de patru biti sunt citite cu MSB la stânga si cu LSB la dreapta. Mai mult, se admite că coeficientul de cel mai înalt ordin al polinomului de intrare este simbolul de patru biti situat cel mai la stânga. Pe baza acestor reguli, de fapt arbitrare, polinomul de intrare este dat de

$$m(x) = x^{12} + \alpha^{14}x^3 + \alpha^{11}x^2 + \alpha^8.$$

Cuvântul de cod produs prin codarea nesistematică este dat de

$$c(x) = m(x)g(x)$$

unde $g(x) = x^2 + \alpha^5 x + \alpha^3$, polinomul minimal pentru rădăcinile α si α^2, în general polinomul minimal pentru puterile până la a $k - 1$ ale primitivei α.
Prin efectuarea multiplicării se obtine

$$c(x) = x^{14} + \alpha^5 x^{13} + \alpha^3 x^{12} + \alpha^{14} x^5 + \alpha^{13} x^4 + \alpha^5 x^3 + \alpha^6 x^2 + \alpha^{13} x + \alpha^{11}.$$

Folosind aceeasi conventie în ordinea reprezentării, cuvântul de cod corespunde următoarei secvențe de biți la iesire:

0001 0110 1000 0000 0000 0000 0000 0000 0000 1001 1101 0110 1100 1101 1110

Pentru calcule se poate recurge la tabelul care urmează, cu atentie la ordinea puterilor care poate fi diferită!

α^k	**Clase de resturi** *modulo* $q(x) = 1 + x + x^4$	**Repre-zentare binară**	α^k	**Clase de resturi** *modulo* $q(x) = 1 + x + x^4$	**Repre-zentare binară**
α^0	1	1000	α^8	$1 + \alpha^2$	1010
α^1	α	0100	α^9	$\alpha + \alpha^3$	0101
α^2	α^2	0010	α^{10}	$1 + \alpha + \alpha^2$	1110
α^3	α^3	0001	α^{11}	$\alpha + \alpha^2 + \alpha^3$	0111
α^4	$1 + \alpha$	1100	α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	1111
α^5	$\alpha + \alpha^2$	0110	α^{13}	$1 + \alpha^2 + \alpha^3$	1011
α^6	$\alpha^2 + \alpha^3$	0011	α^{14}	$1 + \alpha^3$	1001
α^7	$1 + \alpha + \alpha^3$	1101	α^{15}	1	1000

SEMNALE

Lucrarea 9

Tema 1: Serii Fourier

Un semnal priodic de perioadă T_0 se poate dezvolta în anumite conditii conform relatiei

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos n\omega_0 t + b_n \sin n\omega_0 t) .$$

Partea dreaptă a acestei egalităti este o serie, *seria Fourier* a semnalului $s(t)$, iar coeficentii a_n, $n = 0, 1, 2, \ldots$, b_n , $n = 1, 2, \ldots$ se calculează cu formulele

$$a_n = \frac{2}{T_0} \int_{\alpha}^{\alpha + T_0} s(t) \cos n\omega_0 t$$

$$b_n = \frac{2}{T_0} \int_{\alpha}^{\alpha + T_0} s(t) \sin n\omega_0 t$$

Frecventa unghiulară (pulsatia) care apare în expresiile coeficentilor si în seria Fourier este legată de perioadă prin relatia $\omega_0 = 2\pi/T_0$.
Fie acum un semnal periodic de formă rectangulară

$$s(t) = s(t + 2n\pi) = \begin{cases} -1 & -\pi < t \le 0 \\ 1 & 0 < t \le \pi \end{cases}$$

Perioada lui este 2π si amplitudinea 1. Se pot verifica relativ usor următoarele fapte: functia este impară si de aceea coeficientii termenilor în cosinus din

dezvoltarea Fourier sunt toti nuli; coeficientii termenilor în sinus sunt nenuli, dar numai pentru indicii impari si au valorile $4/(n\pi) = 4/[(2k-1)\pi]$, $k = 1, 2, \ldots$. Scriptul Matlab care urmează reconstituie semnalul dintr-un număr finit de componente ale seriei Fourier asociate: 5, 10, 15 sau 20. Graficele obtinute la executarea scriptului arată cât de (im)perfectă este reconstituirea. Diferentele fată de semnalul rectangular originar sunt din ce în ce mai reduse pe măsură ce sunt luate în calcul mai multe componente sinusoidale. Se mentin totusi "falduri" de formă ondulatorie, în zonele unde semnalul originar este de fapt constant, ±1. În vecinătatea momentelor unde semnalul rectangular face saltul abrupt de la –1 la +1 (sau invers) se observă depăsiri ale unitătii pozitive sau negative cu circa 18%. Aceste depăsiri se mentin oricâte componente sinusoidale s-ar lua în considerare. Fenomenul are un nume în literatura de specialitate: *fenomenul Gibbs*.

```
% Program fald
% Recompunerea unui semnal rectangular periodic (impar)
% de perioada 2*pi dintr-un numar finit de componente
% Fourier
% Seria Fourier are toti termenii in cosinus nuli
clear
t=-pi:0.01:pi;    % initializarea unui vector de timp
f=0*t;
subplot(2,2,1)
for k=1:5
   f=f+sin((2*k-1)*t)/(2*k-1);
end    % recompunerea din 5 componente
plot(t,(4/pi)*f)
hold on
grid on
plot([-4 4],[0 0],'k')
plot([0,0],[-2 2],'k')
title('n=5')
f=0*t;
subplot(2,2,2)
for k=1:10
   f=f+sin((2*k-1)*t)/(2*k-1);
end    % recompunerea din 10 componente
plot(t,(4/pi)*f)
hold on
grid on
plot([-4 4],[0 0],'k')
plot([0,0],[-2 2],'k')
title('n=10')
f=0*t;
subplot(2,2,3)
for k=1:15
   f=f+sin((2*k-1)*t)/(2*k-1);
end    % recompunerea din 15 componente
plot(t,(4/pi)*f)
hold on
grid on
```

```
plot([-4 4],[0 0],'k')
plot([0,0],[-2 2],'k')
title('n=15')
f=0*t;
subplot(2,2,4)
for k=1:20
   f=f+sin((2*k-1)*t)/(2*k-1);
end    % recompunerea din 20 de componente
plot(t,(4/pi)*f)
hold on
grid on
plot([-4 4],[0 0],'k')
plot([0,0],[-2 2],'k')
title('n=20')
```

Tema 2: Compensarea fenomenului Gibbs

În unele aplicatii fenomenul Gibbs poate deranja. De aceea este necesar ca într-un fel sau altul el să fie compensat/corectat. Modalitatea cea mai utilizată constă într-o trecere a semnalului printr-un filtru. Scriptul Matlab următor realizează această compensare printr-un filtru *sinc*. Functia **`sinc`** este implementată în Matlab si poate fi înteleasă prin obisnuitul apel **`help sinc`**. Se poate observa diminuarea faldurilor, dar si o pierdere în viteza de trecere de la valoarea –1 la valoarea +1 (si invers).

```
% Program sfald
% Compensarea efectului Gibbs
t=-pi:0.01:pi;    % se initializeaza un vector temporal
f=0*t;
n=10;
% reconstiturea semnalului se realizeaza din 10 componente
for k=1:n
   f=f+sin((2*k-1)*t)/(2*k-1);
end
plot(t,(4/pi)*f)
% se reprezinta reconstituirea fara compensare
hold on
grid on
plot([-4 4],[0 0],'k')
plot([0,0],[-2 2],'k')
f=0*t;
% se corecteaza semnalul reconstituit prin trecerea
% printr-un filtru sinc
for k=1:n
   f=f+sin((2*k-1)*t)*sinc(k/n)/(2*k-1);
end
plot(t,(4/pi)*f,'r')
% se reprezinta reconstituirea dupa compensare
title('COMPENSAREA FENOMENULUI GIBBS LA N=10')
```

Lucrarea 10

Tema 1: Eşantionarea semnalelor; reconstituirea din eşantioane

Scriptul Matlab de mai jos face operatia de eşantionare a semnalului

$$s(t) = \cos(2\pi t) + \cos(4\pi t - \varphi)$$

si de reconstituire a lui din eşantioane. Spectrul semnalului contine numai două frecvente: 1 si 2 Hz. Conditia de aplicare a teoremei eşantionării este indeplinită: semnalul este de spectru/bandă limitat(ă). Este necesar ca frecvența esantioanelor să fie de cel putin 4 Hz, adică cel putin dublul celei mai mari frecvențe din spectru, si semnalul poate fi reconstituit din esantioanele sale.
Scriptul permite câteva intervenții premergătoare derulării calculelor. Se poate modifica faza φ a componentei de 2 Hz, ceea ce modifică forma semnalului (dar nu spectrul lui). Se poate modifica durata semnalului, se poate modifica frecvența prelevării esantioanelor. Prin aceste modificări se pot observa calitatea reconstituirii si magnitudinea efectelor de capăt datorate lipsei unor esantioane premergătoare primului esantion si următoare ultimului esantion.
Se poate experimenta si o frecvență a esantioanelor inferioară celei limită conform cu teorema esantionării. În acest caz semnalul "reconstituit" este altceva decât semnalul originar.

```
% Program esant
% Conditii tipice de executare a "script-ului"
%
% tmax=5        - timpul maxim de reprezentare
% fi=pi/2       - faza componentei secundare
% pasmic=0.001  - pasul utilizat la reprezentarea grafica
% fes=7.0       - frecventa de esantionare
% rep=0         - optiune de reprezentare:
%           (rep=0) grafic unic, (rep>0) grafice separate
tmax=5;
fi=pi/2;
pasmic=0.001;
fes=7.0;
rep=1;
tes=1/fes;     % perioada esantioanelor
t=0:pasmic:tmax;
y=cos(2*pi*t)+cos(4*pi*t-fi);
% pregatirea graficului 1 (semnal)
t1=0:tes:tmax;
n=floor(tmax/tes)+1;
y1=cos(2*pi*t1)+cos(4*pi*t1-fi); % esantionarea
y2=y1(1)*sinc(t/tes);
% pregatirea graficului 2 (reconstituire)
for k=1:(n-1)
    y2=y2+y1(k+1)*sinc(t/tes-k);
end
if rep>0
   subplot(3,1,1)
   plot(t,y)
% trasarea graficului 1 (semnal)
```

```
   hold on
   plot([0 tmax],[0 0])
   for i=1:n
      plot([t1(i) t1(i)],[0 y1(i)],'b:')
% trasarea graficului 2 (reconstituire)
   end
   ylabel('Original')
   title('ESANTIONAREA SEMNALELOR')
   subplot(3,1,2)
   plot(t,y2,'k')
   hold on
    plot([0 tmax],[0 0])
    for i=1:n
    plot([t1(i) t1(i)],[0 y1(i)],'b:')
   end
   ylabel('Reconstituire')
   subplot(3,1,3)
   plot(t,y-y2,'r')
% trasarea graficului 3 (diferente)
   hold on
    plot([0 tmax],[0 0])
   ylabel('Diferente')
   xlabel('Timp(s)')
else
   plot(t,y)
% trasarea graficului 1 (semnal)
    hold on
    plot([0 tmax],[0 0])
    for i=1:n
      plot([t1(i) t1(i)],[0 y1(i)],'b:')
% trasarea graficului 2 (reconstituire)
    end
   plot(t,y2,'k')
   plot(t,y-y2,'r')
% trasarea graficului 3 (diferente)
    plot([0 tmax],[0 0])
   title('ESANTIONAREA SEMNALELOR')
   ylabel
 ('Semnal(albastru)/Reconstituire(negru)/Diferente(rosu)')
   xlabel('Timp(s)')
end
```

Problema 89.

Se dă semnalul periodic

$$s(t) = s(t + nT) = \begin{cases} \cos(100\pi t) & t \in \left[-\frac{T}{4}, \frac{T}{4}\right) \\ 0 & t \in \left[\frac{T}{4}, \frac{3T}{4}\right) \end{cases},$$

pentru orice n întreg, cu $T = 0{,}02$ s. (o tensiune alternativă cu frecvența de 50 Hz si amplitudinea de 1 V, redresată ideal pe alternanța pozitivă, pe o sarcină pur rezistivă de 1 ohm).
Se cer componentele de diverse frecvențe din dezvoltarea Fourier. Aceeasi problemă pentru redresarea ideală dublă alternanță.
Cum se modifică spectrul de componente când elementul redresor ideal se deschide mai târziu cu α, $0 < \alpha < T/2$, față de momentul $-T/4$ (redresare cu comandă pe un tiristor)?
Solutie: În cazul redresării complete a alternanței pozitive, semnalul este o funcție pară de timp, asadar toti coeficientii b_k ai funcțiilor sinus din dezvoltare sunt nuli. Pentru termenii în cosinus, pentru $k = 0, 1, 2, \ldots$, se obtin coeficientii

$$a_k = \frac{2}{T}\int_{-\frac{T}{4}}^{\frac{3T}{4}} s(t)\cos(100k\pi t)dt = \frac{2}{T}\int_{-\frac{T}{4}}^{\frac{T}{4}} \cos(100\pi t)\cos(100k\pi t)dt =$$

$$= 50\int_{-0{,}005}^{0{,}005} \cos[100(k+1)\pi t]dt + 50\int_{-0{,}005}^{0{,}005} \cos[100(k-1)\pi t]dt =$$

$$= \frac{\sin[100(k+1)\pi t]}{2(k+1)\pi}\Bigg|_{-0{,}005}^{0{,}005} + \frac{\sin[100(k-1)\pi t]}{2(k-1)\pi}\Bigg|_{-0{,}005}^{0{,}005} =$$

$$= \frac{\sin[(k+1)\pi/2]}{(k+1)\pi} + \frac{\sin[(k-1)\pi/2]}{(k-1)\pi}$$

În particular, pentru $k = 0$, $a_0 = 2/\pi$, pentru $k = 1$, $a_1 = 1/2$.
Pentru cazul deschiderii cu întârzierea α a elementului redresor, limita inferioară a integralelor de mai sus se modifică în $-0{,}005 + \alpha$.
Pentru redresarea ambelor alternanțe, semnalul trebuie redefinit. El devine

$$s(t) = |\cos(100\pi t)|$$

Perioada se modifică, devine $T = 0{,}01$ s, componenta continuă se dublează, componenta de 50 Hz din dezvoltare se anulează.

Problema 90.

Este cunoscută transformata Fourier a semnalului aperiodic rectangular de amplitudine A si de durată τ, simetric fată de originea timpului:

$$S(\omega) = \int_{-\frac{\tau}{2}}^{\frac{\tau}{2}} Ae^{-j\omega t}dt = A\tau\,\frac{\sin\frac{\omega\tau}{2}}{\frac{\omega\tau}{2}}$$

Folosind acest rezultat si proprietătile transformării Fourier se cere scrierea transformatelor Fourier ale următoarelor semnale aperiodice:

a. Acelasi semnal rectangular întârziat cu timpul α.

Solutie: Întârzierea se traduce în domeniul frecventelor printr-o multiplicare cu $e^{-j\omega\alpha}$. Asadar, transformata cerută este $S(\omega) = A\tau e^{-j\omega\alpha}\dfrac{\sin\dfrac{\omega\tau}{2}}{\dfrac{\omega\tau}{2}}$

b. Semnalul $s(t) = \begin{cases} -1 & t \in [-\tau, 0) \\ 1 & t \in [0, \tau) \\ 0 & t \in R - [-\tau, \tau) \end{cases}$

Solutie: Semnalul este o sumă a doua semnale rectangulare de amplitudini –1, +1 decalate în timp fată de semnalul din enunt cu $-\tau/2$, respectiv cu $+\tau/2$. Transformarea Fourier este o transformare liniară. Asadar

$$S(\omega) = (-1)\tau e^{j\omega\frac{\tau}{2}}\frac{\sin\dfrac{\omega\tau}{2}}{\dfrac{\omega\tau}{2}} + (+1)\tau e^{-j\omega\frac{\tau}{2}}\frac{\sin\dfrac{\omega\tau}{2}}{\dfrac{\omega\tau}{2}} = -2j\sin\frac{\omega\tau}{2}\frac{\sin\dfrac{\omega\tau}{2}}{\dfrac{\omega\tau}{2}}$$

c. Semnalul $s(t) = \begin{cases} 1 & t \in [-2\tau, -\tau) \cup [\tau, 2\tau) \\ 2 & t \in [-\tau, \tau) \\ 0 & t \in R - [-2\tau, 2\tau) \end{cases}$

Solutie: Semnalul este o sumă a două semnale de amplitudine $A = 1$, simetrice fată de originea timpului, unul de durată 4τ, celălalt de durată 2τ. Liniaritatea transformării Fourier si schimbarea lui τ în 4τ, respectin în 2τ produc rezultatul

$$S(\omega) = 4\tau\frac{\sin 2\omega\tau}{2\omega\tau} + 2\tau\frac{\sin\omega\tau}{\omega\tau}$$

d. Semnalul $s(t) = \begin{cases} 1-|t| & t \in [-1,1) \\ 0 & t \in R - [-1,1) \end{cases}$ (functia *cort*)

Solutie: Prin derivarea functiei cort se obtine semnalul de la punctul b), cu $\tau = 1$ si multiplicat cu –1. Reciproc, functia cort se obtine prin integrarea de la $-\infty$ la t a semnalului de la punctul b) particularizat ($\tau = 1$). Operatia de integrare în domeniul timp are ca efect în domeniul frecventelor multiplicarea cu factorul $1/j\omega$. Asadar

$$S(\omega) = 2j\sin\frac{\omega}{2}\frac{\sin\dfrac{\omega}{2}}{\dfrac{\omega}{2}}\frac{1}{j\omega} = \frac{\left(\sin\dfrac{\omega}{2}\right)^2}{\left(\dfrac{\omega}{2}\right)^2}$$

N.B. Transformarea Fourier a derivatei, respectiv a integralei unui semnal se obtine prin multiplicarea sau divizarea cu $j\omega$ a transformatei Fourier a semnalului. Se recomandă ca exercitiu demonstrarea acestei afirmatii.

Problema 91.

Să se calculeze functia de autocorelatie a semnalului

$$s(t) = A\exp(j\omega t)$$

unde A este o constantă.

Solutie: Semnalul este periodic de perioadă $T = 2\pi/\omega$, asa încât functia de autocorelatie este

$$f(\tau) = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} A\exp(j\omega t)A\exp[j\omega(t+\tau)]dt = \frac{A^2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \exp[j\omega(2t+\tau)]dt =$$

$$= \frac{A^2}{2j\omega T}\exp[j\omega(2t+\tau)]\Big|_{-\frac{T}{2}}^{\frac{T}{2}} = \frac{A^2}{\omega T}\frac{\exp(j\omega\tau) - \exp(-j\omega\tau)}{2j} = \frac{A^2}{2\pi}\sin\omega\tau$$

si este o functie periodică de aceeasi perioadă T.

Problema 92.

Să se calculeze functia de autocorelatie pentru semnalul

$$s(t) = A\exp(j\omega t)$$

unde A este o variabilă aleatoare uniform repartizată pe intervalul [0, 1].

Solutie: Pentru o realizare în care A ia valoarea $A^{(k)}$, functia de autocorelatie este cea de la **Problema 91** cu deosebirea că în loc de A^2 apare $[A^{(k)}]^2$. Luând acum media tuturor realizărilor, trebuie pusă în formulă media pătratului variabilei aleatoare A uniform repartizate pe intervalul [0, 1], adică momentul de ordinul 2 al acelei variabile

$$\int_{-\infty}^{+\infty} A^2 f(A)dA = \int_0^1 A^2 1dA = \frac{A^3}{3}\Big|_0^1 = \frac{1}{3}$$

astfel că, în final, functia de autocorelatie este

$$\frac{1}{6\pi}\sin\omega\tau$$

Problema 93.

Se dă semnalul

$$s(t) = A\cos\omega t + B\sin\omega t$$

cu A si B constante. Se cere functia de autocorelatie a semnalului.

Solutie: Semnalul este periodic de perioadă $T = 2\pi/\omega$ asa încât functia de autocorelatie este

$$f(\tau) = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} (A\cos\omega t + B\sin\omega t)[A\cos\omega(t+\tau) + B\sin\omega(t+\tau)]dt =$$

$$= \frac{1}{T}A^2\int_{-\frac{T}{2}}^{\frac{T}{2}} \cos\omega t\cos\omega(t+\tau)dt + \frac{1}{T}B^2\int_{-\frac{T}{2}}^{\frac{T}{2}} \sin\omega t\sin\omega(t+\tau)dt +$$

$$+ \frac{1}{T}AB\int_{-\frac{T}{2}}^{\frac{T}{2}} [\sin\omega t\cos\omega(t+\tau) + \cos\omega t\sin\omega(t+\tau)]dt \quad \text{etc.}$$

O altă posibilitate de calcul trece prin relatiile lui Euler:

$$\cos a = \frac{e^{ja} + e^{-ja}}{2},\ \sin a = \frac{e^{ja} - e^{-ja}}{2j}$$

Prin înlocuire în expresia semnalului se obtine

$$s(t) = A'\exp(j\omega t) + B'\exp(-j\omega t)$$

cu *A'*, *B'* numere complexe. La **Problema 91** a fost evaluată o functie de autocorelatie utilizabilă în problema de fată.

Problema 94.

Determinati functia de autocorelatie a semnalului aperiodic

$$s(t) = A + Bt + Ct^2$$

stiind că *A* si *C* sunt variabile aleatoare independente uniform repartizate pe intervalul [– 1, 1] si *B* este o constantă.

Solutie: Desigur, semnalul în forma dată nu poate fi decât limitat ca durată, altminteri energia ar fi infinită, integralele de evaluat n-ar fi convergente. Se admite asadar că semnalul durează *T*, în particular de la 0 la *T*, în rest fiind nul.

Pentru o realizare particulară

$$s^{(k)}(t) = A^{(k)} + Bt + C^{(k)}t^2$$

functia de autocorelatie este

$$f^{(k)}(\tau) = \int_{-\infty}^{\infty} s^{(k)}(t)s^{(k)}(t+\tau)dt =$$

$$= \int_{\max(0,-\tau)}^{\min(T,T-\tau)} \left[A^{(k)} + Bt + C^{(k)}t^2\right]\left[A^{(k)} + B(t+\tau) + C^{(k)}(t+\tau)^2\right]dt$$

integrala definită a unui polinom de gradul 4 în *t*, a cărui primitivă este imediată. Rezultatul contine termeni ai căror coeficienti sunt sume de termeni de forma $[A^{(k)}]^2$, $A^{(k)}B$, $A^{(k)}C^{(k)}$, B^2, $BC^{(k)}$, $[C^{(k)}]^2$. Medierea pe toate realizările (*k*) ale semnalului va produce medii ale acestor mărimi care sunt respectiv $\int_{-\infty}^{\infty} x^2 f(x)dx = \int_{-1}^{1} x^2\frac{1}{2}dx = \left.\frac{x^3}{6}\right|_{-1}^{1} = \frac{1}{3}$, 0 (deoarece media variabilei *A* este nulă, iar *B* este o constantă), valoarea corelatiei variabilelor aleatoare *A* si *C* (în

particular 0 dacă ele sunt independente), B^2 (ca media unei constante, care este egală totdeauna cu acea constantă), 0 si, din nou, 1/3.