Pázmány Péter Katolikus Egyetem, Információs Technológiai Kar

Digitális jelfeldolgozás nagyfeladat 2006.

ADATTÖMÖRÍTÉS

Az adattömörítés a statisztikus jelfeldolgozás egyik fontos alkalmazása.
A feladat célja a prediktív kódolás bemutatása és szemléltetése

Készítette:

Fodor László **Laki László** **Siklósi Borbála**

Konzulens:
Hegyi Barnabás

2006.június 5.

Tartalomjegyzék

Bevezetés

A számítástechnika és az információs technológia egyre gyorsabb fejlődésének és egyre növekvő igényeinek kielégítése egyre szükségszerűbbé teszi a gyors és könnyű információtovábbítást, illetve a minél nagyobb, összetettebb adatok továbbítását is hasonló gyorsasággal. Ezek a területek nem csak a fejlődést igénylik, hanem egyre szélesebb körben terjednek, a hétköznapi felhasználók körében is, ezért fontos, hogy az összetett adatátvitel mindenki számára megvalósítható legyen. Mivel azonban a szélessávú adatút nem mindenki számára hozzáférhető és drága is, ezért van egyre nagyobb jelentősége az adattömörítés tudományának.

Az adattömörítés a számítógépes tudományágaknak az a területe, melynek célja az adatok feldolgozása oly módon, hogy azok minél kevesebb helyet foglaljanak, vagy minél gyorsabban lehessen őket továbbítani. Ez úgy lehetséges, hogy a valós világ adatai többnyire igen redundánsan és nem a lehető legtömörebb formában reprezentálódnak. Az adattömörítés során igyekszünk a bitek számának csökkentésére a kisebb méretű adatállományok létrehozása érdekében.

Az adattömörítéssel szoros összefüggésben álló területek a kódelmélet és a kriptográfia. Ezekhez az információ-elmélet és az algoritmusos információ-elmélet nyújtanak elméleti hátteret. Amikor azonban az adat tömörítése jelformák alakításaként jelentkezik, gyakran jelfeldolgozási módszereket alkalmazunk.

Adaptív jelfeldolgozás

Az adaptív jelfeldolgozás lényege véletlen jelek manipulációja valamilyen előírt karakterisztika szerint.

Egy gyengén stacionárius és ergodikus jelből indulunk ki, ezt szeretnénk a lehető legjobban közelíteni egy kívánt jelhez, ami szintén stacionárius és ergodikus folyamat. A bemeneti jelre jellemző a korrelációs mátrixa, a kívánt jelet pedig a keresztkorrelációs vektor határozza meg. Az összehasonlítást egy Wiener szűrő (pl módosítható paraméterű FIR szűrő) segítségével hajtjuk végre oly módon, hogy a bemeneti jelet keresztülküldjük ezen a szűrőn, és az így kapott jelet hasonlítjuk össze a referenciával. Így egy különbséget kapunk, ami a hibajel. Célunk a hibajel minimalizálása, azaz várható értékének (átlagának) a négyzetes várható értéke a lehető legkisebb legyen. Kétfélére módszer alkalmazható a szűrő optimális paramétereinek meghatározására:

- Ha ismerjük a korrelációs mátrixot és a keresztkorrelációs vektort, akkor a kiszámolandó a hiba négyzetének várható értékének minimuma, azaz

$$w_{opt}=\min E\,(d_k-y_k)^2=E\,(\varepsilon_k)^2$$

- Ha nem ismerjük a korrelációs mátrixot és a keresztkorrelációs vektort, akkor a következő közelítéssel élünk:

$$\mathbf{w_{opt}=\min \frac{1}{K}\sum_{k=1}^{K}(d_k-x_k)^2}$$

Az első esetben egy szélsőérték meghatározásról van szó, amit gradiensképzéssel oldunk meg, így az eredmény a

$$w_{opt} = R^{(-1)}\, r$$ egyenlet lesz.

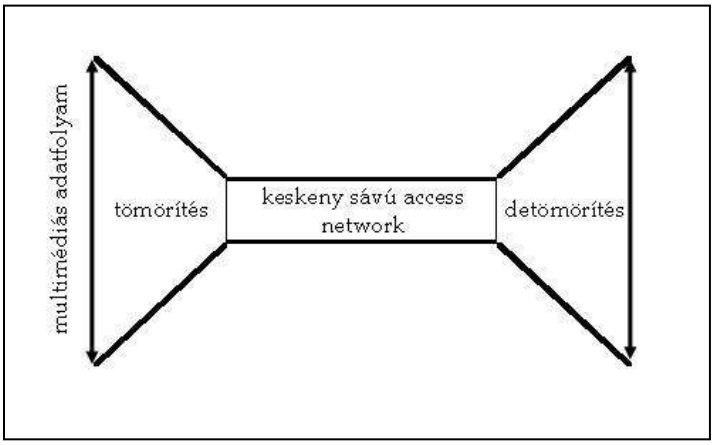
A második esetben problémát okoz, hogy nem ismerjük a szükséges adatokat, ezért van szükség közelítésre. A valós életben ez a gyakoribb, ezért fontos ez a probléma és természetesen, hogy minél pontosabb megoldást találjunk. Az egyik jó közelítési módszert a Robbins – Monroe algoritmus adja meg, miszerint

$$w_l(k+1) = w_l(k) - \Delta\left(d_k - \sum_{j=0}^{J} w_j * x_{k-j}\right) * x_{k-l}$$

a szűrő optimális paramétereinek kiszámítása.

Alkalmazás adattömörítésre

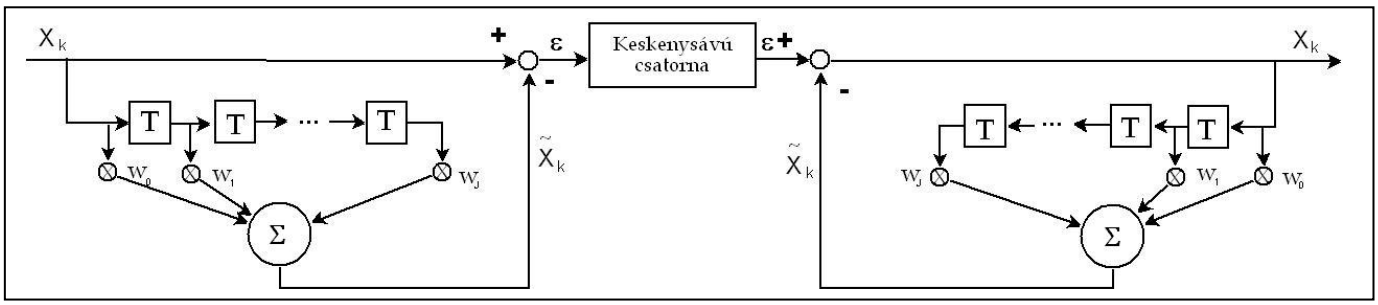
Az információs technológia fejlődése és sokszínűsége következtében az adatátvitel módja igen sokféle lehet. A legelterjedtebb a keskeny sávszélességű access network, ami a legolcsóbb. Létezik közel végtelen sávszélességű backbone network. Az adattömörítés feladata, hogy pl a multimédiás adatfolyamot olyan formába tömörítsük, hogy az keskeny sávszélességű adatúton gyors átviteli sebességgel átjusson, majd detömörítéssel visszanyerjük az eredeti jelet.



A tömörítési eljárások alapvetően két csoportra oszthatók: veszteségmentes és veszteséges tömörítésre. A veszteségmentes eljárással tömörített adatok visszaállítás után azonosak az eredetivel. A veszteségmentes tömörítés használata elengedhetetlen futtatható kódok, szövegfájlok stb. esetében, mert ezeknél akár egyetlen bit megváltozása sem megengedhető. Adaptív predikció során a tömörítés veszteségmentes.

Az adathalmazok reprezentálhatnak pl. képeket vagy feldolgozásra váró egyéb digitalizált jeleket is. Ezekben az esetekben sűrűn előfordul, hogy a tárolás vagy az adatátvitel folyamán nem kell megőrizni az adatok eredeti állapotát. Az ilyen reprezentációk mindig tartalmaznak bizonyos mértékű zajt és torzítást. Ha a tömörítés és visszaalakítás során elvesző adatok a zajt és/vagy a torzítást még éppen elfogadható mértékben növelik csak meg, akkor nyugodtan használhatunk veszteséges tömörítést. A veszteséges tömörítési eljárások sokkal nagyobb tömörítési hatásfokkal dolgoznak, de a tömörítési arány növelésével növekszik a hasznos jelhez adódó zaj is.

Az adattömörítés alapjában véve a korrelációt használja ki. Az alábbi ábrán látszik, hogy az ε_k hibajelet a bemeneti jel és a szűrt jel különbségeként kapjuk, tehát nagyságrendekkel kisebb, mint az eredeti jel, így keskeny sávszélességű csatornán továbbítható.



Tehát:

$$x_k = \tilde{x}_k + \epsilon_k \quad \rightarrow \quad \epsilon_k = x_k - \tilde{x}_k$$

ahol

$$\tilde{x}_k = \sum_{j=1}^{J} w_j x_{k-j}$$

A későbbi problémát az okozza, hogy az átjuttatott hibajelből vissza kell állítani az eredeti jelet.

Fontos szempont, hogy ε_k minél jobban megközelítse az optimális értéket, amit a szűrő együtthatóinak optimalizálásával érhetünk el.

Mivel

$$\bar{w}_{opt}: \min_{w} E(x^2) - 2\sum_{j=1}^{J} w_j E(x_k x_{k-j}) + \sum_{j=1}^{J}\sum_{i=1}^{J} w_i w_j E(x_{k-i} x_{k-j})$$

és

$$w_{opt}: \min_{\bar{w}} R(0) - 2w^T r + w^T R w$$

valamint

$$E\left(x_k^2\right) >> E\left(\epsilon_k^2\right)$$

Így a fenti megállapítások és a Robbins Monroe algoritmus felhasználásával a következő eredményt kapjuk:

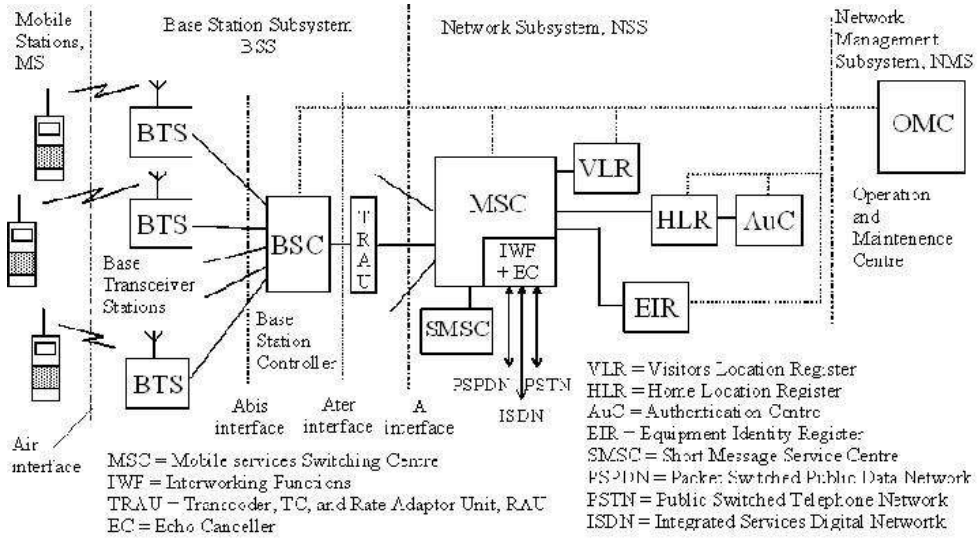
$$E\left(\epsilon_k^2\right) = R(0) - 2\,\bar{w}_{opt}^T \bar{r} + \bar{w}_{opt}^T R \bar{w}_{opt} = R(0) - \bar{w}_{opt}^T R \bar{w}_{opt} = E\left(x_k^2\right) - \bar{w}_{opt}^T R \bar{w}_{opt}$$

GSM szabvány és az adaptív prediktív kódoló

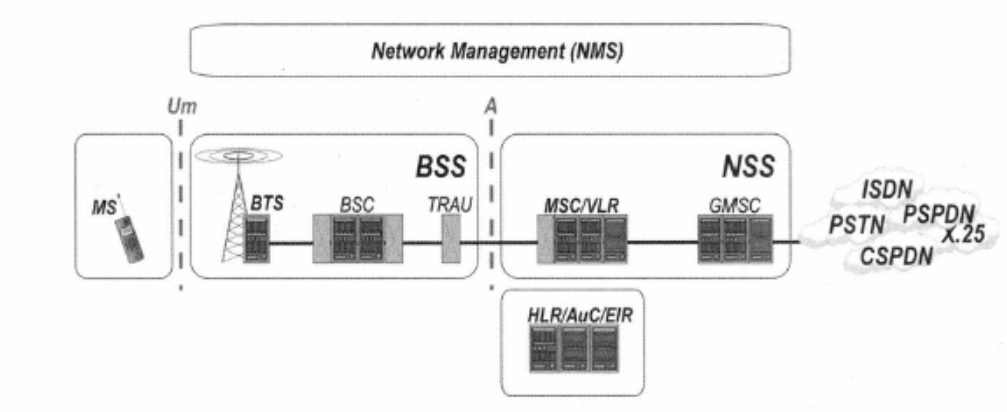
A lineáris prediktív kódoló egyik gyakorlati alkalmazása a GSM szabványban rögzített.

1991-ben a létrejött az első GSM rendszer (Global System for Mobile Communication), melynek eredeti célja a digitális technológián alapuló mobilkommunikáció szabvány a 900 MHz-es sávra, mely biztosítja a globális perszonális kommunikációt. Még 1991-ben definiálták a Digital Cellular System 1800-at, valamint az USA-ben a Personal Communicaton System 1900-at. A következő évben számos európai GSM hálózat jött létre.
Manapság több mint 1,8 milliárd ember használja 210 országban.
Ezen rendszerek fő elemei: a mobil állomás és a GSM hálózat. A mobil állomás egy mobil berendezésből és egy előfizetői azonosító modulból (SIM-Subscriber Identity Modul) áll.

A GSM rendszer architektúrája:



Decentralizált implementáció:



MS: Mobile Station

BSS: Base Station Subsystem

NSS: Network Subsystem

NMS: Network Management Subsystem

GSM frekvenciák:

uplink: 890 – 915 Mhz	(GSM)
downlink: 935 – 960 MHz	(GSM)
uplink: 1710 – 1785 MHz	(DCS 1800)
downlink: 1805 – 1880 MHz	(DCS 1800)
uplink: 876 – 880 MHz	(GSM-R)
downlink: 921 – 925 MHz	(GSM-R)

A sávok 200KHz-es vivőfrekvenciákra vannak bontva, így 124 (GSM) és 374 (DCS 1800) vivő van. Minden egyes vivő 8 fizikai csatornára van osztva (TDMA), így a frekvencia csatornák száma: kb. 1000 GSM és 3000 DCS 1800.

Beszéd Csatorna:

A GSM lineáris prediktív kódolót (13kbps) (**LPC** – Linear Predictive Coding) alkalmaz. Az LPC célja a bitek számának csökkentése. Főleg audio jelek és beszéd feldolgozásra használják. Ez az egyik legerőteljesebb beszéd analizáló technika és az

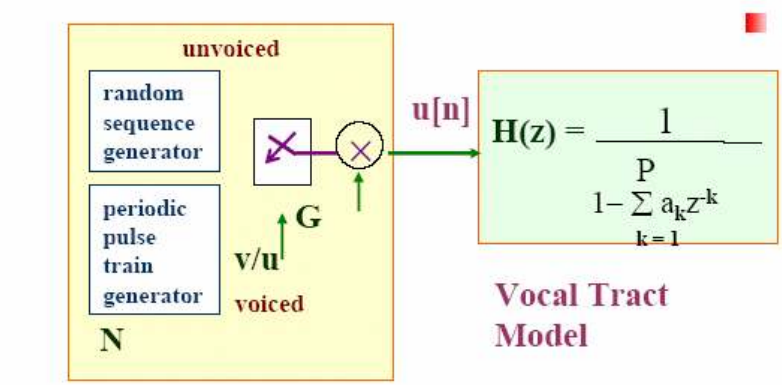
egyik leghasznosabb metódus jó minőségű beszéd tömörítésre alacsony bitszámmal, valamint rendkívül pontos becslő paramétereket ad a beszédre.
Iniciálisan olyan közelítést használ, mely szerint a beszéd jelet egy 'berregő' adja ki egy cső végén. Bár ez elég durvának tűnik, de a valóságban jól alkalmazható a beszédre. Az LPC analizálja a beszédet, úgy hogy megbecsüli a 'formantokat' (csúcs egy akusztikus frekvencia spektrumban), kiveszi hatásukat a jelből (inverse filtering), megbecsüli az intenzitását és frekvenciáját a megmaradó zúgásnak = buzz (residue).
Az LPC úgy szintetizálja a jelet, hogy megfordítja a folyamatot: a zúgás paramétereit és a residue-t használja egy jelforrás létrehozásához, a formantokat használja egy szűrő létrehozásához, aztán végigfuttatja a jelet a szűrőn, és ez eredményezi majd a beszédet.
Mivel a beszéd jelek időben nem állandóak, ezért ezt a procedúrát framenként végzik. Általában 30-50 frame/sec ad értelmes beszéd tömörítést.

LPC koefficiens:

Az LPC-t gyakran spektrális jelburkoló átvitelre használják, és ezért átviteli hibákra toleránsnak kell lennie. A szűrő együtthatók közvetlen továbbítása nem elvárható, mivel nagyon érzékenyek a hibákra. Más szóval, egy nagyon kis hiba is képes teljeses eltorzítani a teljes spektrumot, vagy ami még rosszabb, egy kis hiba instabillá teheti a szűrőt.
Léteznek fejlettebb reprezentációk, mint pl. log area ratios (LAR), line spectral pairs (LSP).
Az LPC beszéd modellje: auto-regresszív model:

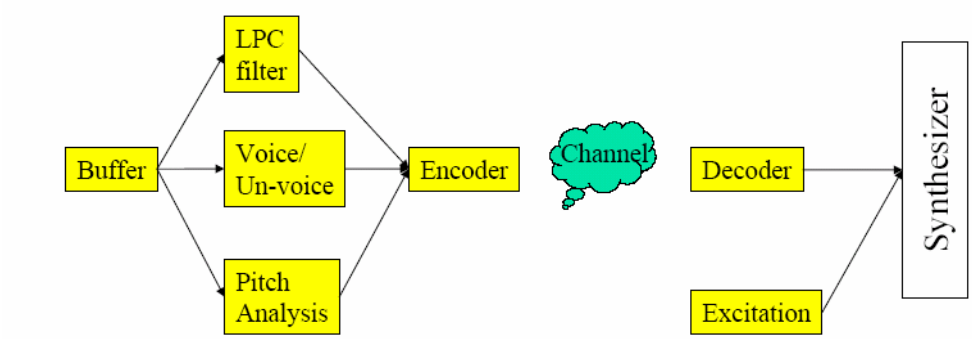
$$s(n) = \sum_{k=1}^{p} a(k)s(n-k) + e(n)$$

A paramétereket {a(k); k=1:p} megkaphatóak egy Toeplitz egyenletrendszer megoldással.



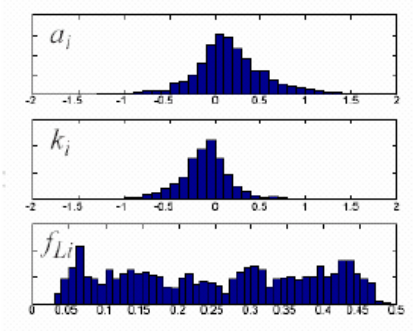
Ha beszédet szeretnénk tömöríteni, akkor a kvantált paramétereket {a(k)}-t, és G-t kell átvinni.

Az LPC struktúrája:



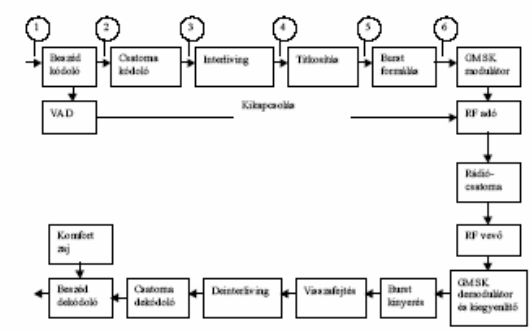
Ahhoz hogy a minőség kommunikációra alkalmas legyen, 8 KHz-es mintavételezést (4Khz-es Bandwith), 10. rendű LPC-t használnak, és 20-30 ms-ként (300-500 param/sec) update-elik.

Reprezentálás és kvantálás:



Ahol, {ki} a reflekciós együttható, az fLi-nél pedig a log area ratios (LAR)-t látjuk, ami azt mutatja, hogy stabil.

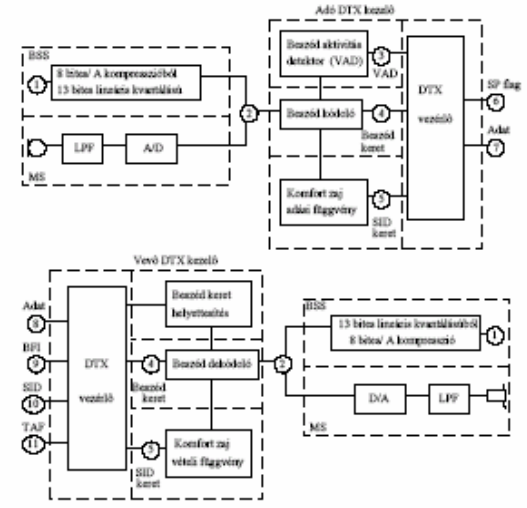
A beszéd forgalmi csatorna átviteli útja:



Adatsebességek:

1 Beszédkódoló bemenete : 104 kbit/s
2 Beszédkódoló kimenete : 13 kbit/s
3 Csatornakódoló kimenete : 22.8 kbit/s
4 Interliving után : 22.8 kbit/s
5 Titkosítás után : 33.8 kbit/s
6 Burst adatsebessége : 270 5/6 kbit/s

Beszéd kódolás és dekódolás menete:



(1) 8 bites/ A törvényü kompresszióval kvantált PCM jel (CCITT G.711 ajánlás), 8000 minta/sec

(2) 13 bitre lineárisan kvantált PCM, 8000 minta/sec

(3) Beszéd aktivitás flag

(4) Kódolt beszéd keret, 50 keret/sec, 260 bit/keret = 13 kb/sec

(5) SID (Silence Descriptor) keret, Kódolt háttérzaj minta, 260 bit/keret

(6) Beszéd flag = jelzés, hogy az adatbitek beszéd vagy SID információt hordoznak

(7) A rádiós alrendszerhez továbbított adat bitek

(8) A rádiós alrendszer által vett adat bitek

(9) BFI Hibás keretet indikáló flag

(10) SID jelző flag

(11) TAF jelzőflag, mely az SID keret helyzetét mutatja a SACCH multikereten belül

Csatorna kódolás és interleaving:

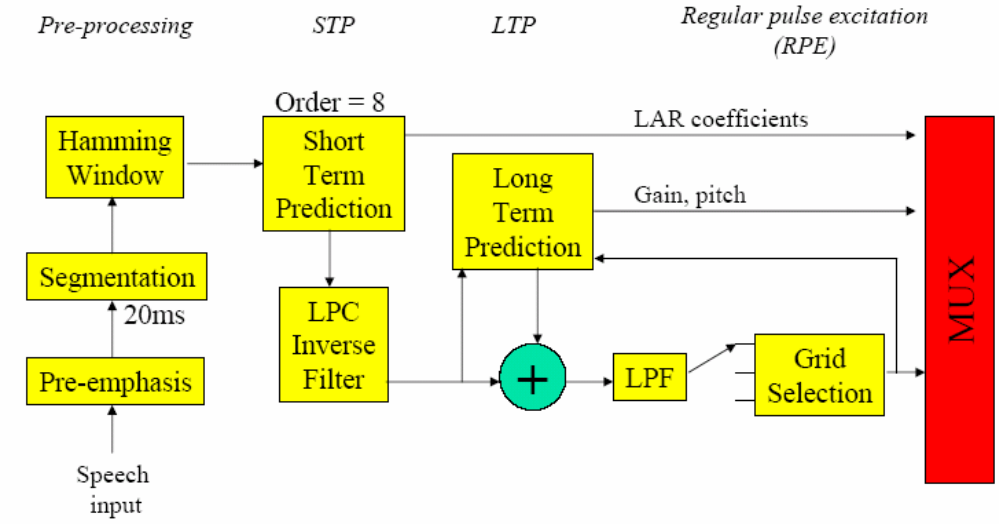
A csatorna kódolás szerepe a hibák kijavítása, melyeket a csatorna okozott az információ átvitelekor. Ehhez olyan (FEC) hibajavító kódokat használnak, ahol a source információs bitjeit egyéb bitekkel egészítik ki, így a transfer alatt keletkezett hibák detektálhatók és reparálhatók. A GSM rendszernél blokk és konvolúciós kódokat használnak. Ezek bizonyos elfordulási hibák javítására alkalmasak, de a burst (csomós) hibákra nem alkalmazhatóak. Mivel egy mobil rádiócsatorna fadingjei (fakulás) az esetek többségében hiba burst-öket okoznak, ezért a csatorna kódolást interleavinggel (bitsorrend átrendezés) kapcsolják össze.

Az információs bitsorozatok átrendeződnek, így a hibaburst-ökből (csomókból) különálló hibák keletkeznek, melyek a csatorna kódolással javíthatók. Fontos, hogy az interleaving nem növeli az átvivendő adatmennyiséget.

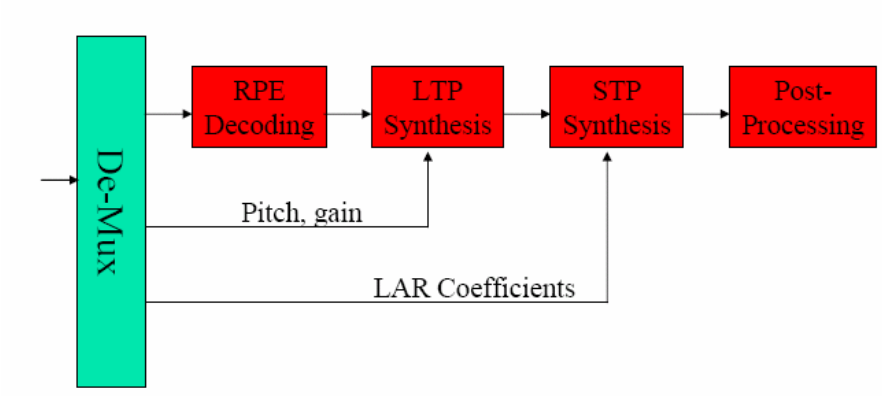
V.42bis Adattömörítő software:

Ez egy olyan software, mely a hibajavító modemek teljesítményét növeli azáltal, hogy jobb kihasználtságot biztosít egy gyakran túltelített kommunikációs csatorna számára. A fix hosszúságú karakterek megjelenítésére változtatható hosszúságú kódot használ, ahol a leggyakrabban átvitt karakterek tömörítve lesznek és egy olyan kódszóval reprezentálódnak, melyek sokkal rövidebbek a konvencionális bit kódnál. Egy ilyen rendszer egy szótárt alkalmaz, melyben egy hierarchikus fa-struktúrában tárolódnak a gyakran előforduló karakter stringek a kódjukkal egyetemben, melyek ezeket a karakter stringeket reprezentálják.

GSM beszéd tömörítés folyamata:



De-tömörítése:



A feladat

Készítsen MATLAB függvényt, amely demonstrálja az adattömörítést! Generáljon egy
normális eloszlású és adott korrelációjú n x véletlen 4 10 hosszú jelsorozatot, ha a korrelációs mátrix a következő:

$$\mathbf{R} = \begin{bmatrix} 1 & 0.5 & 0.2 & 0.4 & 0.05 \\ 0.5 & 1 & 0.5 & 0.2 & 0.4 \\ 0.2 & 0.5 & 1 & 0.5 & 0.2 \\ 0.4 & 0.2 & 0.5 & 1 & 0.5 \\ 0.05 & 0.4 & 0.2 & 0.5 & 1 \end{bmatrix}$$

A legenerált x_n jelsorozat első feléből becsülje az **R** mátrixot és **r** vektort, majd Wiener szűrés feladatát megoldva, adja meg az optimális FIR szűrő együtthatókat!

Adja meg az ε_n tömörítési hiba jelsorozatot! Becsülje, majd hasonlítsa össze az x_n bemenő és az ε_n hiba jelfolyam szórását!

Készítsen adaptív prediktív kódolót!

A megadott hosszúságú és korrelációs függvényű normális eloszlású sztochasztikus jelsorozat előállítása:

```
x = normrnd(zeros(1,10000),ones(1,10000));
```

A FIR szűrő együtthatói a következő algoritmussal számíthatók:

$$x_n = \sum_{i=0}^{4} a_i \nu_{n-i}$$

$$R(k) = E\{x_n x_{n-k}^*\} = E\left\{\left(\sum_{i=0}^{4} a_i \nu_{n-i}\right)\left(\sum_{j=0}^{4} a_j \nu_{n-j-k}\right)^*\right\} = \sum_{i=0}^{4}\sum_{j=0}^{4} a_i a_j^* E\{\nu_{n-i}\nu_{n-j-k}^*\} = \sum_{i=0}^{4}\sum_{j=0}^{4} a_i a_j^* \delta_{i,j+k} =$$

$$= \sum_{i=k}^{4} a_i a_{i-k}^*$$

ahol $k = 0,1,2,3,4$ **és** * **konjugálás művelete.**

Az R mátrix 0.4 és 0.2 értékeit a könnyebb és szemléletesebb kezelés érdekében felcseréltük. A feladatban megadott eredeti mátrixhoz tartozó FIR szűrő átviteli függvényének együtthatói csak komplex értékűek lettek.

A fenti számítást Mathematica programmal a következő módon adott eredményt:

```
NSolve[{a^2+b^2+c^2+d^2+e^2==1, a*b+b*c+c*d+d*e==0.5, a*c+b*d+c*e==0.4, a*d+b*e==0.2, a*e==0.05},
 {a, b, c, d, e}]
```

A több megoldás közül ezzel a valós értékűvel számolunk tovább:

```
h=[-0.13344142,-0.52216354,-0.75371082,-0.032578176,-0.37469625];
```

A jelsorozat szűrése:

```
L=50;
Xn=filter([h],[1],x);
S=[zeros(1,L)];
Se=[zeros(1,L)];

for J = 1:1:L
```

Az eredeti R korrelációs mátrixnak RE a jel első felének alapján történt becslése:

```
[W,RE]=corrmtx(Xn(1:5000),J);
```

Az r becslése (az RE első oszlopa):

```
r=RE(2:length(RE),1);
```

A Wiener szűrő optimális együtthatóinak kiszámítása ismerve a korrelációs mátrixot és a keresztkorrelációs vektort:

```
    Rv=RE(1:(length(RE)-1),1:(length(RE)-1));
    wopt=Rv^(-1)*r;
```

Szűrés:

```
    Yn=filter(wopt,[1],Xn);
```

A hiba jelsorozat meghatározása:

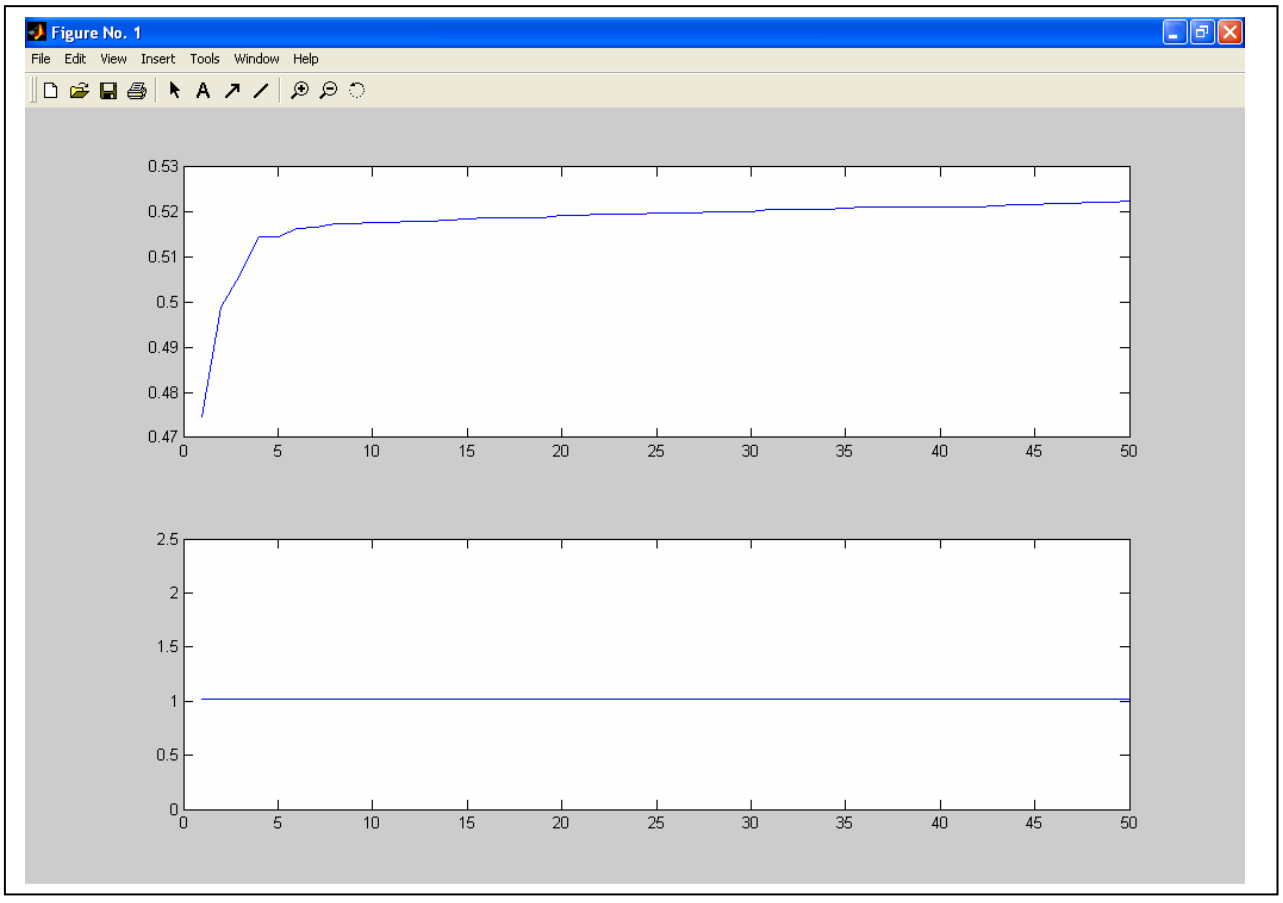
```
    En=Yn-Xn;
```

A hiba jelsorozat szórása:

```
    s=std(En);
    S(J)=s;
    Se(J)=std(Xn);
 end
```

Az ábrákon a hiba szórása és az eredeti jel szórása a J függvényében látható. Függően attól hogy milyen J-t választunk, az En szórása változik. Az első ábrán látszik, hogy egy J-t bizonyos érték után nincs értelme növelni.

```
figure(1)
subplot(2,1,1)
plot(S)
subplot(2,1,2)
plot(Se)
```



A wopt közelítése Robbins-Monroe algoritmussal:

```
 for J=1:1:50
     Z=5000;
 w=(zeros(J,Z));
 Xb=(zeros(1,Z+J));
 Yb=(zeros(1,Z));
 Eb=(zeros(1,Z));
 bu=(zeros(1,J));
 delta= 0.05;
 Sb=(zeros(1,J));

 for c=1:1:10^4
     Xb(J+c)=Xn(c);
 end
 for n=1:1:Z
     for g=1:1:J
        bu(g)=w(g,n)*Xb((n+J)-g);
     end

     Yb(n)=sum(bu);
     Eb(n)=Yb(n)-Xb(n+J);


     for g=1:1:J
         if Z > n
            w(g,n+1)=w(g,n)-delta*(Xb(n+J)-Yb(n))*Xb((n+J)-g);
         end
     end
 end
end
```