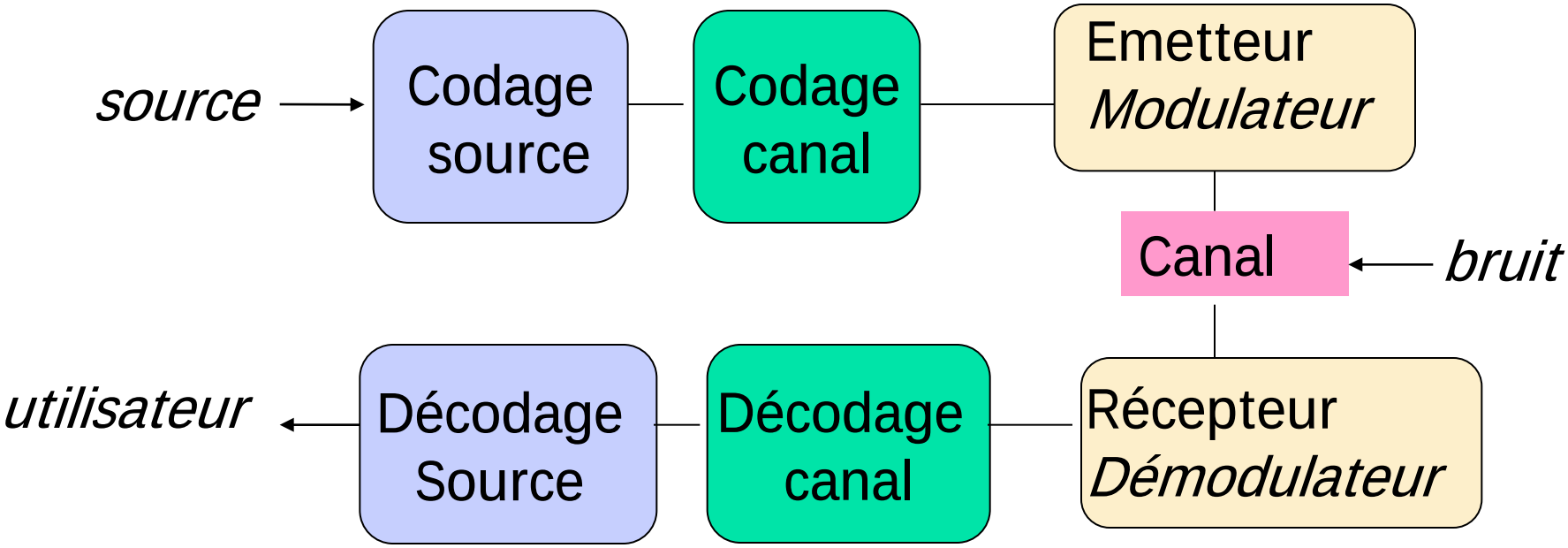
# Système de communications (TNT, 3G, WIFI,...)



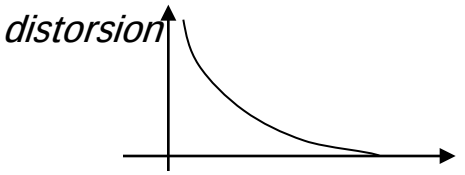

- Source : parole, musique, images, vidéo
- Canal : radio, cable, fibre optique, CD
- Modulateur / Démodulateur (FC1) : adaptation données numériques ↔ canal
- Codage source : compression des données pour une meilleure efficacité
- Codage canal : protection des données pour une meilleure fiabilité

# Théorie de l'information et codage

- I) Codage de source:
  - Généralités
  - Codage entropique (sans pertes)
- II) Codage canal
  - Généralités
  - Information mutuelle – capacité – Théorème du codage canal
  - Codes en blocs – codage de Hamming
  - Codes convolutifs – algorithme de Viterbi

# Codage de source - Généralités

- Objectif principal
  - Meilleure efficacité possible: minimiser le débit binaire à transmettre
- Deux classes d'approches
  - Pour les sources continues (voix, images, musique) : codage avec pertes car il est nécessaire de quantifier les données. Dans ce cas il y a un second objectif qui est de minimiser la distorsion entre les données originales de la source et les données reconstruites pour l'utilisateur
    - Compromis débit/distorsion




    - Idées :
      - Minimiser la distorsion pour les données les plus probables (quantification optimale)
      - Opérer des transformations préalables pour ne quantifier que les données « innovantes» (non prédictibles) ou quantifier avec plus de précision les données les plus « perceptibles » (basses fréquences)
  - Pour les sources discrètes (texte, données déjà quantifiées) : codage sans pertes (distorsion nulle) qui s'appuie sur la notion d'entropie (information moyenne) de la source (**codage entropique**)
    - Idée : moins de bits à transmettre pour les données les plus probables
    - C'est l'idée du MORSE: **E** : **.** **Y** : **-.--**

# Codage entropique

- Exemple
- Information - Entropie
- Code préfixe - Inégalité de Kraft
- Théorème du codage de source
- Code de Huffman
- Source étendue
- Codage arithmétique

# Exemple

- On considère une source X à valeurs dans {Rouge, Vert, Bleu, Jaune}
  de loi 1 {1/4, 1/4, 1/4, 1/4} ou loi 2 {0.4,0.3, 0.2, 0.1}
- Code A: R → 00, V → 01, B → 10, J → 11
  Code B: R → 0, V → 10, B → 110, J → 111
  Code C: R → 0, V → 1, B → 10, J → 01
- Le code C n'est pas décodable: 0110 → RVVR ou RVB ou JB...
- Longueur moyenne des autres codes
  - pour le Code A, on trouve $l_{moy} = 2$ dans les deux cas (loi 1 ou 2)
  - pour le Code B :
    - avec la Loi 1 : $l_{moy}=(1+2+3+3)/4 = 2.25$
    - avec la Loi 2 : $l_{moy}= 1*0.4+2*0.3+3*0.2+3*0.1= 1.8$
- Le meilleur code dépend de la loi de la source
- Peut on trouver un code meilleur que B pour la source X de loi 2?

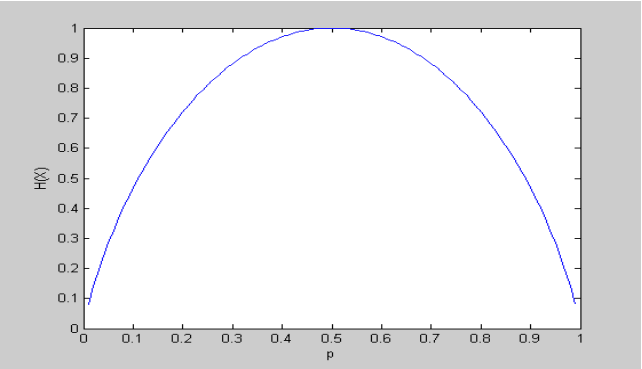
# Notion d'information

- La quantité d'information obtenue lorsque l'évenement X = x se réalise est liée à l'incertitude sur cet évènement: si un évènement est certain, sa réalisation ne nous apporte aucune information, si un évènement est impossible, sa réalisation apporte une information infinie.
  - I(x) = f(p(x)) avec f fonction positive et décroissante
- L'information apportée par deux évènements indépendants doit être la somme des informations liées à chacun des évènements.
  - I(x,y)= f(p(x,y)) = f(p(x)p(y)) = f(p(x)) +f(p(y))
  - $\Rightarrow$ I(x) = $\log_b(1/p(x))$ base b arbitraire
- I(x) = $\log_2(1/p(x))$ : information en bit (binary unit)
- Un bit représente la quantité d'information fournie par le choix d'une alternative parmi deux équiprobables.

- La théorie de l'information, dont on reparlera pour le codage canal est basée sur les travaux de C. Shannon dont l'article de référence est: "A mathematical theory of communications" paru dans en 1948

# Information moyenne - Entropie

- On considère une source discrète X à valeurs dans $A = \{a_1,a_2,...,a_M\}$ de loi $\{p_1,p_2,...,p_M\}$ . L'entropie de cette source, notée H(X), est l'information moyenne:
  - $H(X)=\Sigma_{i=1:M}\, p_i \log_2(1/p_i)$
- Propriétés:
  - L'entropie est maximale si les évènements $X=a_i$ sont équiprobables. $H(X) \leq \log_2(M)$
  - L'entropie est minimale (et nulle) si l'un des évènements $X=a_i$ est certain
- Exemple : pour 2 symboles de probabilités p et 1-p

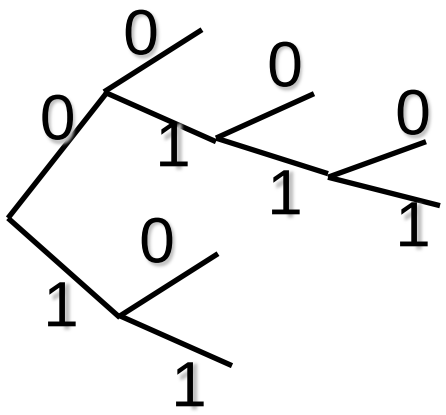


# Retour au Codage

- On considère une Source discrète X (sans mémoire) à valeurs dans $A=\{a_1,a_2,...a_M\}$ de loi $\{p_1,p_2,...,p_M\}$
- Le codage pour cette source consiste à définir des mots de code binaires pour chacun des symboles possibles.
  - $a_i \rightarrow c_i=(b_{i,1},...,b_{i,li})$ : mot de code de longueur $l_i$ associé au symbole $a_i$
- Deux objectifs:
  - Avoir un code à décodage unique et instantané
  - Minimiser la longueur moyenne du code : $l_{moy}=\Sigma_i p_i l_i$

# Décodage unique – code préfixe - Inégalité de Kraft

- Pour être à décodage unique et instantané le code doit remplir la condition du préfixe: aucun mot de code ne doit être le début d'un autre mot de code.
- Un code est dit préfixe s'il remplit cette condition.
- Un code préfixe peut être représenté par un arbre binaire (CNS)
  - Les mots de code sont les suites de 0 et 1 sur les branches allant de la racine jusqu'aux feuilles de l'arbre
  - exemple
    - 00
    - 010
    - 0110
    - 0111
    - 10
    - 11




- Il existe un code préfixe dont les M mots de code sont de longueur $l_1,\ldots,l_M$ si et seulement si:
  - $\Sigma_i 2^{-l_i} \leq 1$ : Inégalité de Kraft

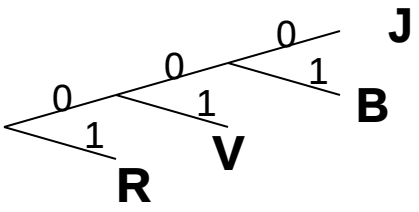
# Théorème du codage de source

- La longueur moyenne d'un code uniquement décodable pour une source discrète sans mémoire X d'entropie H(X) vérifie:
  - $l_{moy} \geq H(X)$
- Il existe un code préfixe (code de Shannon) tel que:
  - $H(X) \leq l_{moy} < H(X)+1$

- Théorème de Shannon : Pour toute source discrète il existe un code dont la longueur moyenne est arbitrairement proche de l'entropie

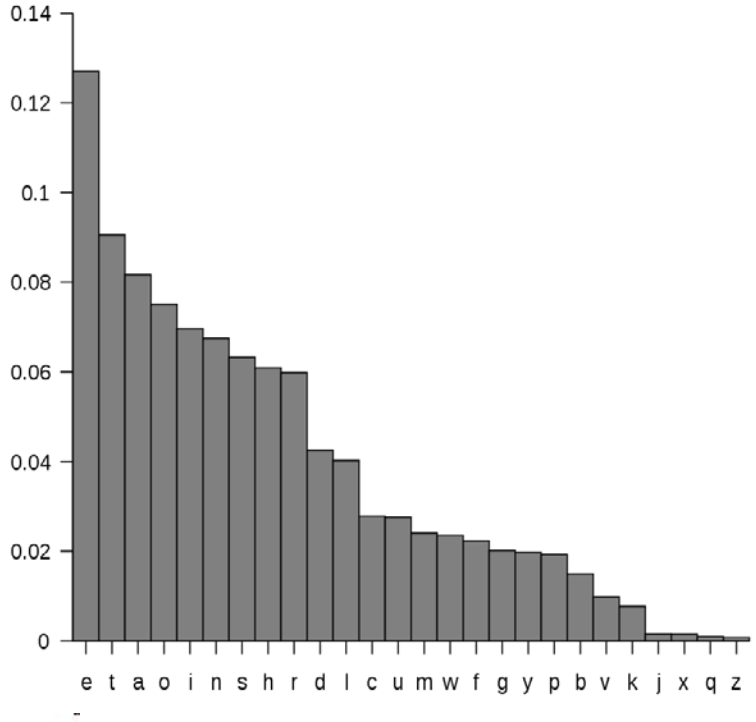
# Code de Huffman

- L'algorithme de Huffman construit l'arbre binaire du code pour une source discrète
- Tant qu'il reste des symboles:
  - On classe les symboles par ordre de probabilité décroissante
  - On sélectionne les 2 symboles de probabilités les plus faibles
  - On crée 2 branches de l'arbre se terminant par ces symboles: ces deux branches sont étiquetées 0 et 1 respectivement
  - A la racine de ces 2 branches, on crée un symbole fictif de probabilité égale à la somme des 2 probabilités
- Le code de Huffman est optimal (il n'existe pas de code préfixe dont la longueur moyenne soit inférieure à celle obtenue par l'algorithme)
- Exemple: {R,V,B,J} de loi {0.4, 0.3, 0.2, 0.1}

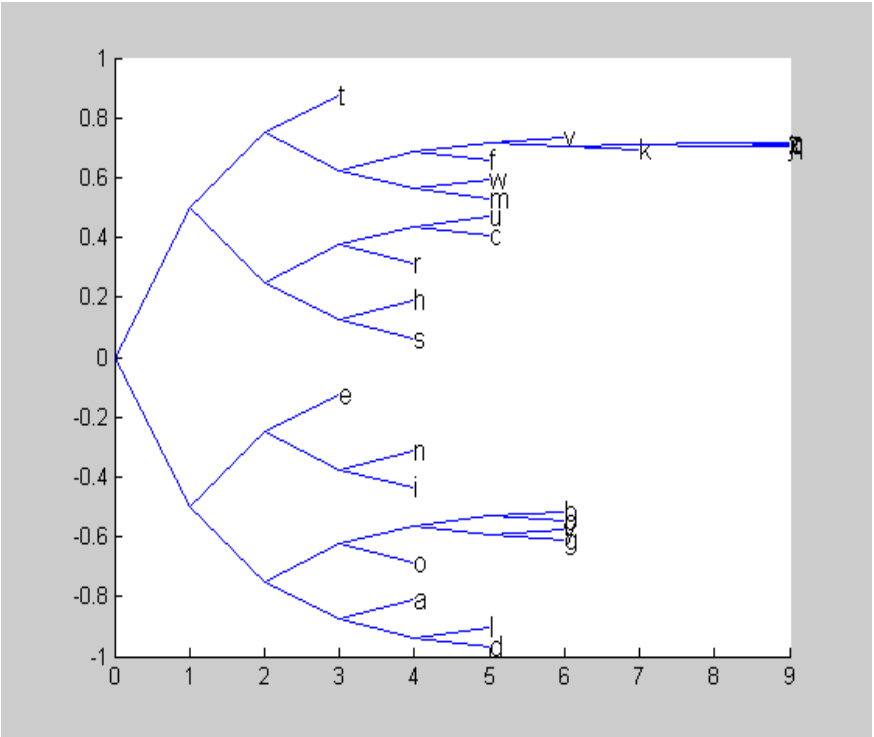


# Exemple de l'alphabet (application : transmission des SMS)

- Distribution de probabilité des lettres (en anglais)
- Entropie associée : H=4.17




- Code de Huffman (Longueur moyenne du code : $\bar{l} = 4.205$ )



# Source étendue à N symboles

- On considère des N-uplets de N symboles successifs : $X^{(N)}=(X_1,X_2,...,X_N)$ avec $X_n$ à valeurs dans $A=\{a_1,a_2,...a_M\}$
- Il y a $M^N$ N-symboles $a^{(N)}$
- Il existe un code préfixe (Code de Shannon) tel que:
  - $H(X_N) \leq l^{(N)}_{moy} < H(X_N)+1$
  - Pour source indépendante, sans mémoire: $H(X_N)=NH(X)$
  - $H(X) \leq l_{moy} < H(X)+1/N$
- Exemple: source X à valeurs dans {Noir, Blanc} de loi {1/10,9/10}
  - Code de Huffman pour la source (N=1)
  - Code de Huffman pour la source N-étendue avec N=3

# Codage arithmétique

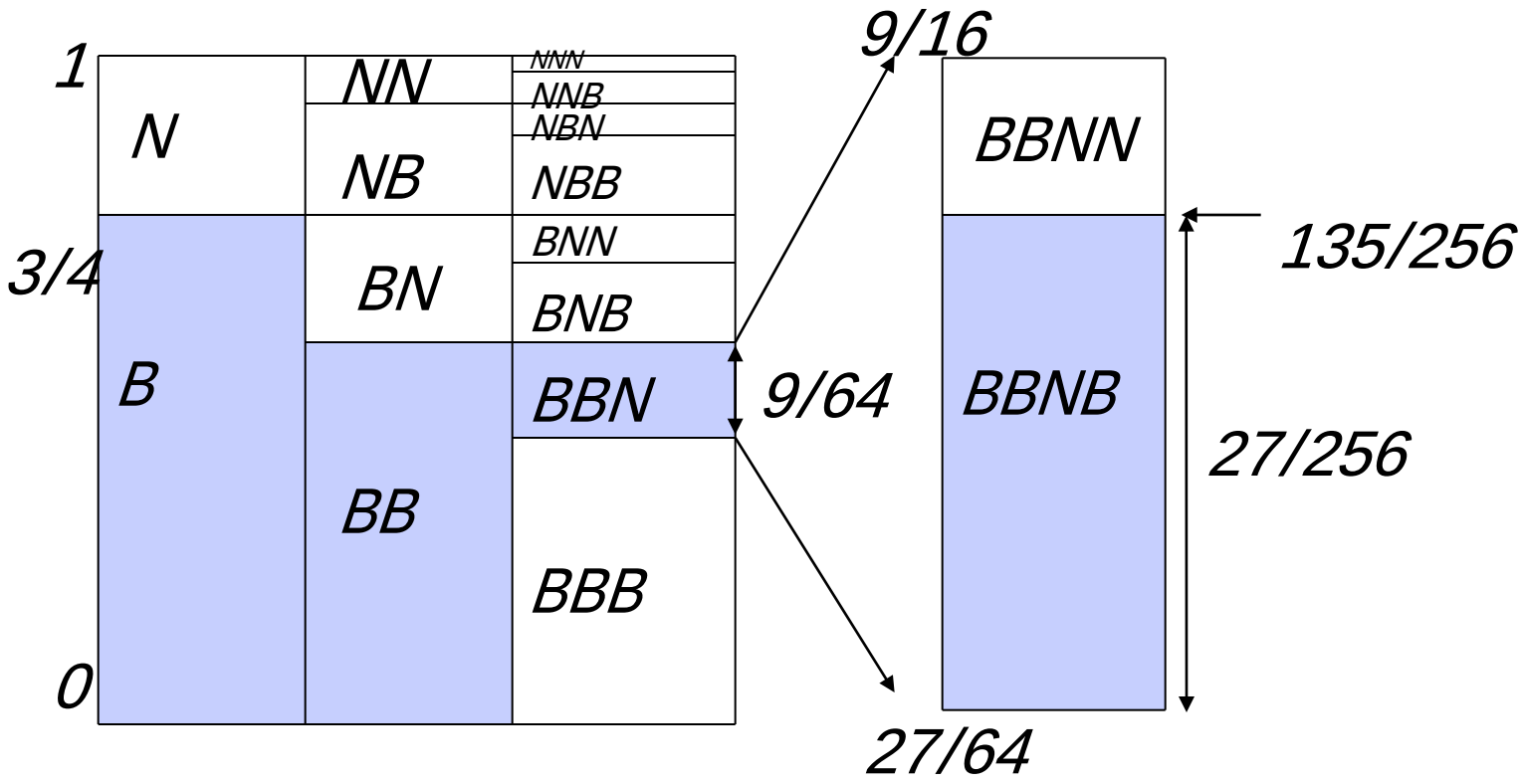
- PRINCIPE: une séquence S de symboles est traduite par un intervalle de [0,1[ dont la longueur est égale à la probabilité de S: cet intervalle est codé sur $l = \lceil \log_2(1/p(S)) \rceil$ bits
- Ce codage est optimal (au sens d'une source N-étendue) pour des séquences de N symboles
- Cet algorithme est en particulier utilisé dans les nouvelles normes de compression d'images
- Bibliographie: excellent article (très détaillé pour mise en œuvre dans « vrai » codeur):
  - Introduction to Arithmetic Coding - Theory and Practice de Amir Said

# Algorithme de codage

- Notations
  - Probabilités : $p_m = p(a_m)$ pour m=1:M
  - Probabilités cumulées : $c_m = \Sigma_{i=1:m-1}\, p_i$ pour m=1:M ,
    cas particuliers : $c_1 = 0, c_{M+1} = 1$
- Condition initiale: $IC^{(0)} = [0,1[$
  - $B_{IC}^{(0)} = 0$ (base) , $L_{IC}^{(0)} = 1$ (longueur)
- Tant qu'il reste des symboles $X_n$ à transmettre:
  - Si $X_n = a_m$, l'intervalle courant $IC^{(n)}$ est réajusté suivant:
    - $B_{IC}^{(n)} = B_{IC}^{(n-1)} + c_m * L_{IC}^{(n-1)}$
    - $L_{IC}^{(n)} = p_m * L_{IC}^{(n-1)}$
  - Remarques:
    - l' intervalle $IC^{(n)}$ s'emboîte dans $IC^{(n-1)}$
    - $L_{IC}^{(n)} = \Pi_{i=1:m}\, p_i$
- Codage:
  - $IC^{(N)}$ est codé par un nombre binaire $(b_1 2^{-1} + \ldots + b_{NB} 2^{-NB})$ qui appartient à cet intervalle: il faut et il suffit que $NB = \lceil \log_2(1/L_{IC}^{(N)}) \rceil$

# Exemple

- 2 symboles B et N de probabilité 3/4 et 1/4
- Séquence BBNB



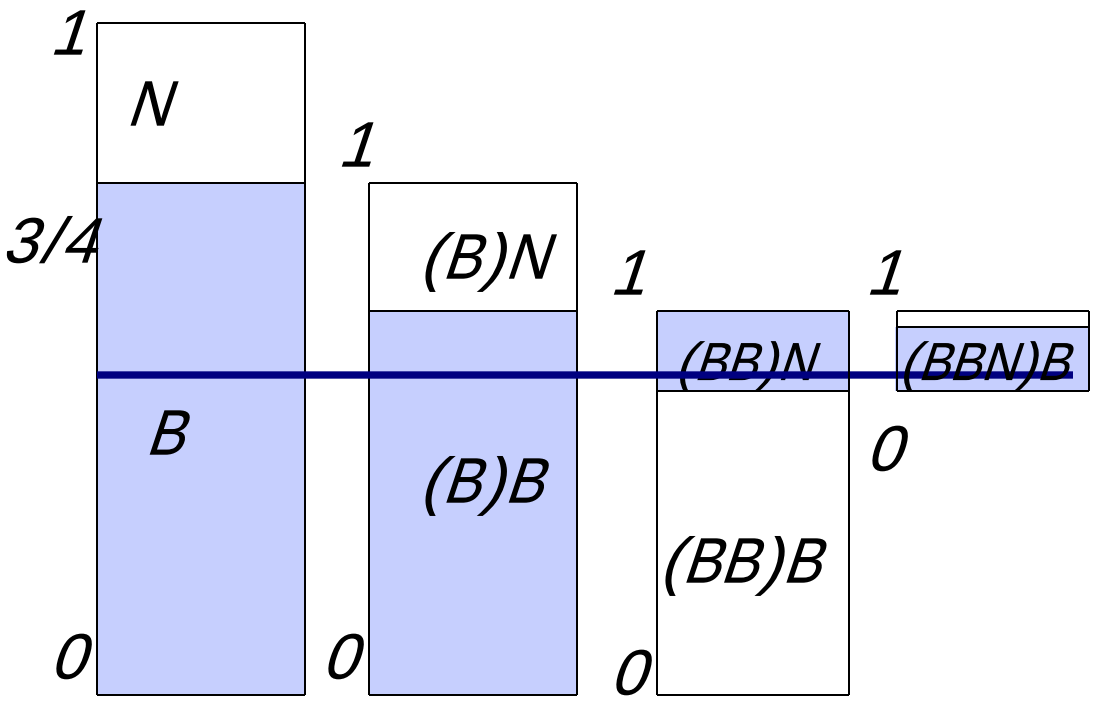

- Code: nombre compris entre 27/64=011011 et 135/256: 10000111
  sur NB= $\lceil \log_2(1/L(IC)) \rceil = \lceil \log2(256/27) \rceil$ = 4 bits
- On peut prendre **0111 (7/16)** ou **1000 (8/16)**

# Algorithme de décodage

- Le code reçu est traduit en un nombre courant NC $\in$ [0,1[
- Tant qu'il reste des symboles à décoder
  - Détermination de la valeur m telle que NC$\in$[$c_m$ $c_{m+1}$[
    - $\Rightarrow$ le symbole «$a_m$ » est décodé
  - Mise à jour du nombre courant : NC= (NC-$c_{m-1}$)/$p_m$
    - $\Leftrightarrow$ l'intervalle [$c_m$ c $_{m+1}$[ est dilaté en [0 1[
    - $\Leftrightarrow$ l'influence du symbole $a_m$ que l'on vient de décoder est annulée
- Remarques
  - Le décodeur doit connaître le vecteur des $p_m$ ou des $c_m$ ainsi que le nombre de symboles à décoder
  - La séquence décodée est exacte quel que soit le code correspondant à un nombre dans l'intervalle $IC^{(N)}$

# Exemple

- Code=0111 : NC=7/16
- NC $\in$ $[c_1\, c_2\, [$
  - $X_1$=B ($=a_1$)
  - NC=NC/(3/4)=7/12
- NC $\in$ $[c_1\, c_2\, [$
  - $X_2$=B
  - NC=NC/(3/4)=7/9
- NC $\in$ $[c_2\, c_3\, [$
  - $X_3$=N ($=a_2$)
  - NC=(NC-3/4)/(1/4)=1/9
- NC $\in$ $[c_1\, c_2\, [$
  - $X_2$=B



# Optimalité du codage arithmétique

- Nombre de bits pour coder N symboles:
  - $NB = \lceil \log_2(1/L_{IC}^{(N)}) \rceil = -\Sigma_{k=1:N} \log_2 p(s_k) + \Delta$ avec $\Delta < 1$
- Nombre de bits moyen par symbole
  - $l_{moy} = E(NB/N) = H(X) + \Delta/N$
  - La longueur moyenne pour un code arithmétique tend vers l'entropie de la source lorsque N croît.
- Pour un code optimal:
  - $l_{moy}$ (nb bits/symb) = H(X) (nb shannon/symb)
  - $\Rightarrow$les bits O et 1 sont équiprobables
  - $\Rightarrow$ les valeurs codées suivent une loi uniforme dans [0,1[
  - La fonction de répartition de ces valeurs est linéaire entre 0 et 1

# Codage canal

- **Généralités**
- Théorie de l'information:
  - Information mutuelle
  - Capacité
  - Théorème du codage canal
- Codes en blocs
  - codage de Hamming
- Codes cycliques
- Codes convolutifs
  - algorithme de Viterbi

# Généralités:
# Contexte - enjeux

- Données originales (émises ou stokées) : 0100101011
- Données (données reçues ou lues) : 01**1**010**0**011
  - Causes de l'altération : mauvaises conditions de réception (téléphone mobile) ou dégradation du support (CD rayé).
  - Résultat : les données reçues sont différentes des données émises.
- L'équipement qui reçoit les données ne "sait" pas qu'il y a des données erronées.
- Ces erreurs ont un impact variable suivant les applications: fort impact pour des données confidentielles ou sensibles (numéro carte visa, commandes militaires) et plus faible impact pour l'altération de paquet de bits dans une conversation téléphonique ou un flux vidéo.

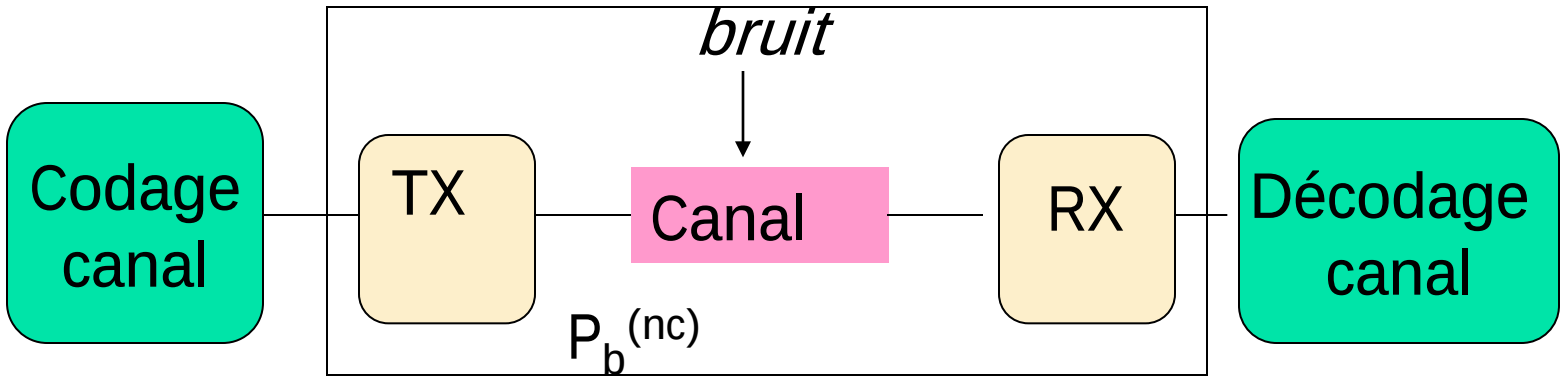
- La première étape consiste à détecter la présence d'erreurs.

# Stratégies en cas d'erreur

- Corriger les erreurs : FEC (Forward Error Correction) protocoles de la couche physique.
- Solliciter l'émission de nouvelles données (protocoles des couches réseau).
  - Mécanisme ARQ (Automatic Repeat reQuest).
- Interpoler entre les données précédentes et les données suivantes (couche application).
  - Technique possible si les données sont redondantes (parole, vidéo) et que la perte d'un paquet ou trame ne dégrade pas trop la qualité de service.

# Détection et Correction d'erreur

- Principe: le codeur (canal) ajoute des bits de redondance aux bits d'information et le décodeur exploite cette redondance pour détecter et éventuellement corriger les erreurs.




- Sans codage canal , cas d'une communication BPSK
  - $P_b^{(nc)} = Q((2E_b/N_0)^{1/2})$

# Exemple: code à répétition C(3,1)

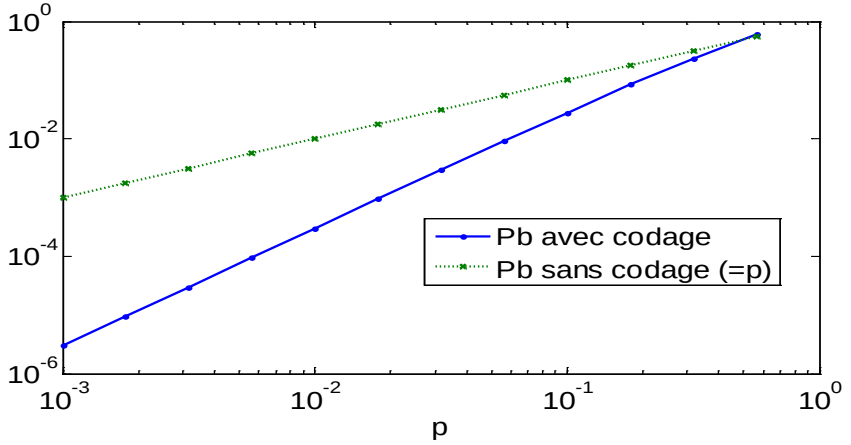
- Chaque bit d'information est répété 3 fois

*010011* → *Codage canal* → *000 111 000 000 111 111*

- Les blocs codés autorisés sont 000 et 111
- Le décodeur prend la décision majoritaire:
  - 0 si deux ou trois 0 dans le mot reçu , 1 sinon

*001 111 010 011 111 101* → *Décodage canal* → *010111*

- Probabilité d'erreur après codage/décodage si $P_b^{(nc)}=p$
  - $P_b^{(c)}=p^3+3p^2(1-p)$



# Limites et inconvénients du codage

- Si le canal est très bruité, le processus de décodage peut dégrader les performances (augmenter la probabilité d'erreur bit).
- La protection croît avec la redondance mais a un coût:
  - La redondance augmente le nombre de bits à transmettre
    - En théorie, l'accroissement du débit qui en résulte augmente l'occupation spectrale et diminue donc l'efficacité.
    - En pratique, le débit est fixé mais le **débit utile** diminue.
  - On appelle rendement du code le rapport :
    (Nb bits utiles)/(Nb bits émis)

# Historique

- 1948 : Shannon (théorie de l'information).
- 1950 : Hamming.
- 1950-1970 : codes en blocs et codes cycliques, BCH (Bose-Chaudhuri-Hocquenghem) et RS (Reed-Solomon).
- 1960-1970 : codes convolutifs (Fano, Forney, Viterbi).
- 1980 : modulations codées en treillis (Ungerboeck).
- 1990 : décodage iteratif et turbo-codes (Berrou-Glavieux).
- 2000 : codes LDPC (Low Density Parity Check).

# Codage canal

- Généralités
- **Théorie de l'information:**
  - Information mutuelle
  - Capacité
  - Théorème du codage canal
- Codes en blocs
  - codage de Hamming
- Codes cycliques
- Codes convolutifs
  - algorithme de Viterbi

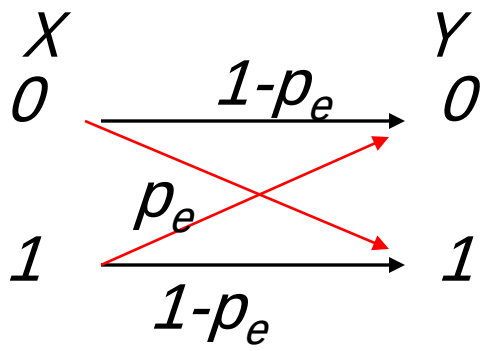
# Théorie de l'information

- Information mutuelle entre deux évènements X=x et Y=y
  - Permet de mesurer combien la réalisation de l'un des deux modifie l'incertitude (l'information) de l'autre.
    - Information de (X=x) : $I(x)=\log_2(1/p(x))$
    - Information de (X=x sachant que Y=y) : $I(x/y) = \log_2(1/p(x/y))$
  - La différence entre les deux informations liées à ces évènements sera l'information mutuelle entre x et y :
    - $I(x,y) = I(x)-I(x/y) = \log_2(p(x/y)/p(x))$
      - Si $p(x/y)>p(x)$ $I(x,y)>0$
      - Si $p(x/y)<p(x)$ $I(x,y)<0$
      - Si $p(x/y)=p(x)$ évènements indépendants $I(x,y)=0$
    - On remarque que : $I(x,y)=\log_2(p(x,y)/(p(x)p(y)))=I(y,x)$

# Information mutuelle moyenne - capacité

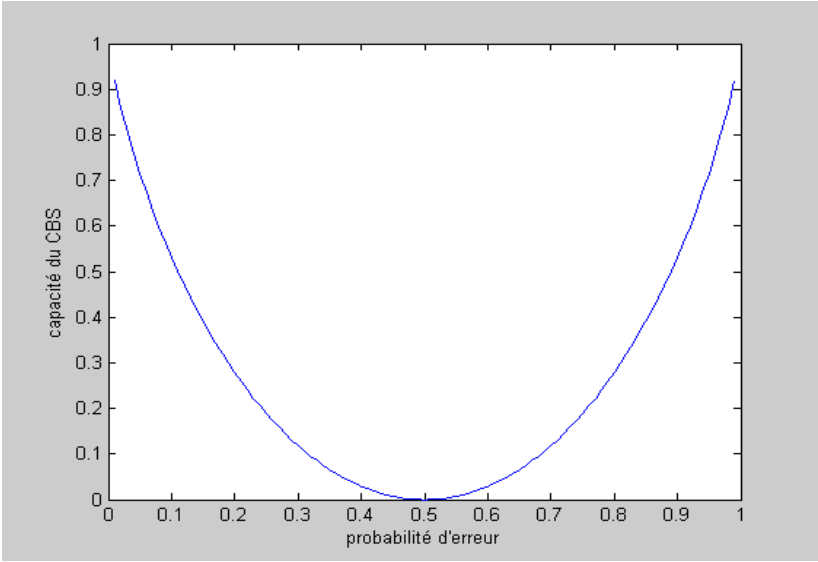
- Pour deux v.a. X et Y discrètes, l'information mutuelle moyenne est définie par:
  - $I(X,Y)=\Sigma_{x,y}p(x,y)I(x,y)= \Sigma_{x,y}p(x,y)\log_2(p(x,y)/(p(x)p(y)))$
- Théorème: $I(X,Y)\geq 0$
- Lien avec l'entropie:
  - Entropie conditionnelle:
    - $H(X/Y=y)= \Sigma_x p(x/y)\log_2(1/p(x/y))$
    - $H(X/Y)= \Sigma_y p(y)\ H(X/Y=y)$
  - $I(X,Y)=H(X)-H(X/Y) = H(Y)-H(Y/X)$
- Lien avec un canal de transmission: X → *canal* → Y
  - Si $H(X/Y) = 0$ $I(X,Y)=H(X)$ : le canal est excellent (connaissant la sortie on n'a plus aucune incertitude sur l'entrée)
  - Si $H(X/Y) = H(X)$ $I(X,Y)= 0$ : le canal est très mauvais (la connaissance de la sortie n'apporte aucune information sur l'entrée)
- La capacité d'un canal est l'information mutuelle maximale qu'il peut transmettre:
  - $C=\max_X I(X,Y)$
  - le maximum est pris par rapport à la loi sur les données X i.e. : H(X) max

# Capacité du canal binaire symétrique (CBS)



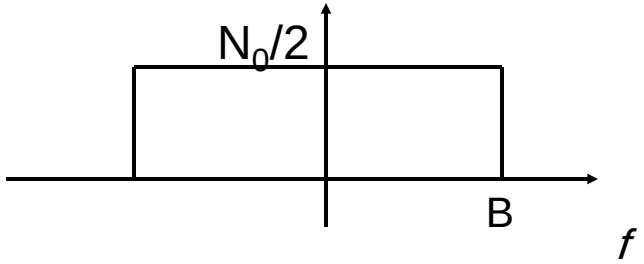

- Canal caractérisé par sa probabilité d'erreur $p_e = P(Y=0/X=1) = P(Y=1/X=0)$
- $H(X)=1 \Rightarrow H(Y)=1$
- $H(Y/X) = -p_e \log_2 p_e - (1-p_e)\log_2(1-p_e)$
  - Ne dépend que du canal

- Capacité $C=1+p_e \log_2 p_e + (1-p_e)\log_2(1-p_e)$



# Capacité d'un canal gaussien à bande limitée

- Pour un canal à bruit additif gaussien : Y = X + N
  - X : données émises de puissance $P_X$ dans une bande limitée B
  - Y : données reçues
  - N : bruit de densité spectrale de puissance $N_0/2$ dans la bande, $\sigma_N^2=N_0B$



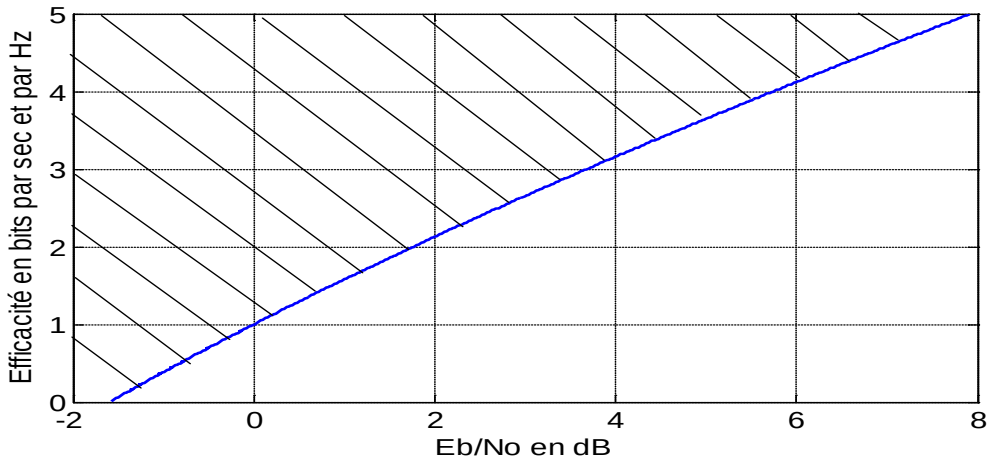

- La capacité de ce canal est:
  - $C=1/2 \log_2(1+P_X/N_0B)$ en bits/échantillon
  - $C=B \log_2(1+P_X/N_0B)$ en bits/sec

# Théorème du codage canal

- Shannon (48): il est possible de transmettre sans erreurs si le débit d'information est inférieur à la capacité du canal
  - La démonstration s'appuie sur des séquences aléatoires de longueur tendant vers l'infini
  - Ce théorème ne donne pas la méthode pour atteindre cette capacité
  - Avant les turbo-codes et les LDPC les techniques de codage sont restées très en deça de cette limite théorique.
- Exemple pour CBS:
  - si $p_e$=.1 C≈1/2
  - Avec un code de rendement 1/2 il doit être possible de transmettre sans erreur
  - La technique du code à répétition présentée page 24 permet d'atteindre $P_b \approx 3.\ 10^{-2}$ pour un rendement 1/3

# Exemple du canal à bande limitée à bruit additif blanc gaussien (BL, BABG)

- On s'intéresse au cas limite : débit (bits/s) = C
  - $P_X$ (énergie/sec)=$E_b$(énergie/bit) C(bits/s)
  - Efficacité spectrale (en bits/sec/Hz) : $C/B=\log_2(1+E_b/N_0 * C/B)$



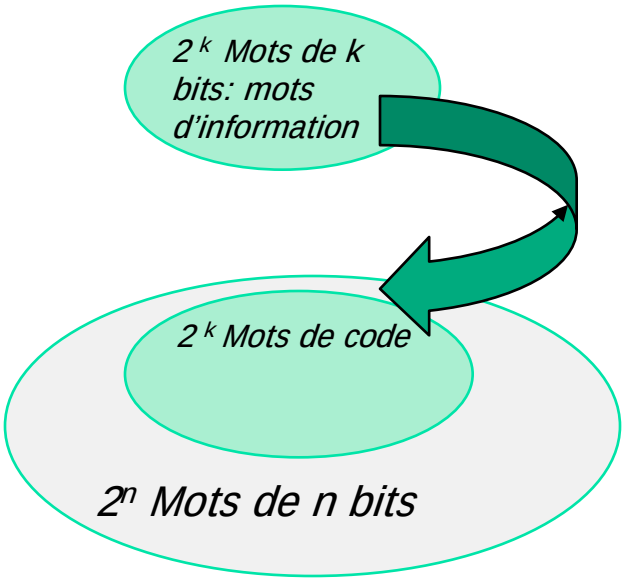

- On peut « théoriquement » transmettre 1 bit/s/Hz avec $E_b/N_0$=0 dB
- Soit : 1000 bits/s sur 1000 Hz de bande

# Codage canal

- Généralités
- Théorie de l'information:
  - Information mutuelle
  - Capacité
  - Théorème du codage canal
- **Codes en blocs**
  - codage de Hamming
- Codes cycliques
- Codes convolutifs
  - algorithme de Viterbi

# Codes en blocs binaires linéaires

- Association de mots de code de n bits à des mots d'information de k bits : codes C(n,k)
  - chaque bloc de n bits en sortie du codeur dépend (linéairement) du bloc de k bits en entrée




  - La somme (modulo 2) de deux mots de code est un mot de code
  - Le mot 00……0 (n bits) est toujours un mot de code
  - Un code est un S.E.V. de dim k de l'espace vectoriel $V_n$ des mots de n bits

# Capacité de Détection d'erreurs : r=c+e

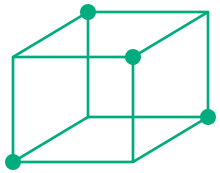
- $2^k$ mots de code c (de n bits) parmi $2^n$ mots dont $2^k$ -1 ≠0
- $2^n$-1 mots d'erreurs non nulles possibles parmi lesquelles $2^k$ -1 ne sont pas détectables car si e=c', r=c+c'=c'' est encore un mot de code.
- $2^n$ - $2^k$ mots d'erreurs détectables
- Le rapport ($2^k$ -1)/($2^n$ - $2^k$) entre le nombre d'erreurs non détectables sur celui des erreurs détectables devient très faible si n-k (le nombre de bits de redondance) croît.

# Poids d'un mot - distribution de poids
# Distance de hamming et distance minimale du code

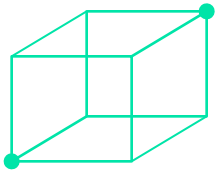
- Poids d'un mot: nombre de bits non nuls
  - w(01100101) = 4
- Distribution de poids: ensemble des poids des mots de code
  - 000 011 110 101 w={0,2}
- Distance de hamming entre deux mots: nombre de bits différents = poids du mot différence (modulo 2)
  - d(01100101,0100010) = 3 = w(00000111)
- Distance minimale du code $d_{min}$: distance de hamming minimale entre deux mots de code = poids du mot de code (non nul) de plus faible poids
  - 000 011 110 101 $d_{min}$ =2

# Principe du codage « classique »

- Les $2^k$ mots de code dans l'ensemble des $2^n$ mots de n bits doivent être les plus distants les uns des autres (au sens de la distance de hamming). Ainsi s'il se produit des erreurs (mais pas trop) pendant la transmission le décodeur pourra détecter ou corriger ces erreurs **en décodant le mot de code le plus proche du mot reçu**.
- Si la distance minimale entre 2 mots est $d_{min}$ on peut détecter toute erreur de moins de $d_{min} - 1$ bits
  - Ex C(3,2) : 00 → 000 , 01 → 011, 10 → 101, 11 → 110 : $d_{min} = 2$



- Si la distance minimale entre 2 mots est $d_{min} = 2t+1$ on peut corriger jusqu'à t erreurs par mot de n bits.
  - Ex C(3,1) : 0 → 000, 1 → 111 : $d_{min} = 3$



# Matrice génératrice d'un code C(n,k)

- Mot d'information : $b = (b_1\ b_2 \ldots b_k)$
- Mot de code associé : $c = (c_1\ c_2 \ldots c_n)$
  - $c_j = \Sigma_{i=1:k}\ g_{ij} b_i$ pour j=1...n
  - $c = b\ G$ avec $G = (g_{ij})$ matrice (k,n)
- G est la matrice génératrice du code
  - Les k lignes de G forment une base du code (S.E.V.)
- Exemples :
  - Code de parité C(3,2) : G = ?
  - Code de répétition C(3,1) : G = ?
  - Code C(5,2) :
    - Mots de code ?
    - Distance minimale du code ?

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

# Décodage par tableau standard (pour faciliter la recherche du mot le plus proche)

- mot reçu r=c+e  ou c : mot de code, e:mot d'erreur
- Tableau standard (tous les mots de n bits dans un tableau de $2^k$ colonnes et $2^{n-k}$ lignes)
- Exemple

| *Mots de code* → | | | |
|---|---|---|---|
| 00000 | 10110 | 01101 | 11011 |
| 00001 | 10111 | 01100 | 11010 |
| 00010 | 10100 | 01111 | 11001 |
| 00100 | 10010 | 01001 | 11111 |
| 01000 | 11110 | 00101 | 10011 |
| 10000 | 00110 | 11101 | 01011 |
| 00011 | 10101 | 01110 | 11000 |
| ~~00101~~ | ~~10011~~ | ~~01000~~ | ~~11110~~ |
| 01010 | 11100 | 00111 | 10001 |

*Mots avec 1 erreur*

*Mots avec 2 erreurs*

# Code systématique

- Un code est dit systématique si les k premiers bits du mot de code sont identiques aux k bits du mot d'information.
  - Les k premiers bits sont dits systématiques et les n-k bits restants sont appelés bits de parité. **Lorsqu'il n'y a pas d'erreur le décodage consiste à récupérer ces k premiers bits**.
  - La matrice génératrice d'un code systématique est sous la forme:

$$G = \begin{pmatrix} 1 & 0 & \dots & 0 & p_{11} & p_{12} & \cdots & p_{1(n-k)} \\ 0 & 1 & \ddots & \vdots & p_{21} & p_{22} & \cdots & p_{2(n-k)} \\ \vdots & \ddots & \ddots & 0 & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & p_{k1} & p_{k2} & \cdots & p_{k(n-k)} \end{pmatrix}$$

- Il est toujours possible d'obtenir la matrice génératrice de la version systématique d'un code par permutation des colonnes de G (ce qui ne modifie pas le poids des mots) et combinaisons des lignes (qui sont des mots de codes).
  - Exemple de la page précédente

# Code dual d'un code C(n,k)

- Soit un code C(n,k) de matrice génératrice G.
- Le code dual est formé des mots orthogonaux aux mots de code. C'est un s.e.v. de dim n-k.
- $V_n = C(n,k) \oplus C^{\perp}(n,n-k)$
- La matrice génératrice H du code dual est de dimensions (n-k,n) et est telle que $GH^T=0$
- Pour un code systématique généré par $G=[I_k \,|P]$ la matrice génératrice du code dual est : $H = [P^T|I_{n-k}]$
  - Exemple : H pour G déterminé page précédente

# Intérêt de la matrice H – décodage par syndrome

- Le mot c de n bits est un mot de code $\Leftrightarrow$ $c H^T = 0$
- Un mot de code a un poids minimal $d_{min}$ $\Leftrightarrow$
  toute famille de $d_{min}$ -1 colonnes de H est libre
- Borne du singleton:
  - au plus n-k colonnes de H libres $\Leftrightarrow$ $d_{min} \leq n-k+1$
- Syndrome pour le mot reçu r=c+e : $s = r H^T$ (de taille n-k)
  - s = 0 s'il n'y a pas d'erreur
  - $s \neq 0$ s'il y a une erreur différente d'un mot de code
  - Il y a $2^{n-k} - 1$ erreurs différentiables (car $2^{n-k} - 1$ mots non nuls différents de taille n-k)
- Exemple : pour G et H définis pages précédentes
- Pour une erreur simple
  - $e_i$ = (0 0...1 ... 0) : une seule erreur en i-ème position
  - $s = e_i H^T$ = i-ème ligne de $H^T$ = i-ème colonne de H

# Code de Hamming C($2^m$ -1, $2^m$ - m – 1)

- Corrige toute erreur simple dans un mot de n bits
  - Les n syndromes s = e $H^T$ = i-ème colonne de H pour i =1 à n doivent être distincts
- Les n colonnes de H doivent être toutes différentes et il y en a $2^{n-k}$ -1 possibles.
- Si on pose m = n-k, n= $2^m$ -1 et k = $2^m$ - m -1
- Construction des codes de Hamming:
  - On construit H dont les colonnes sont les nombres de 1 à n écrits sous forme binaire sur m bits
  - On permute les colonnes de H de façon à l 'écrire sous forme systématique : H = [$P^T$|$I_{n-k}$ ]
  - On détermine G suivant : G=[$I_k$ |P]

# Exemple : C(7,4)

- Matrice de contrôle:

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

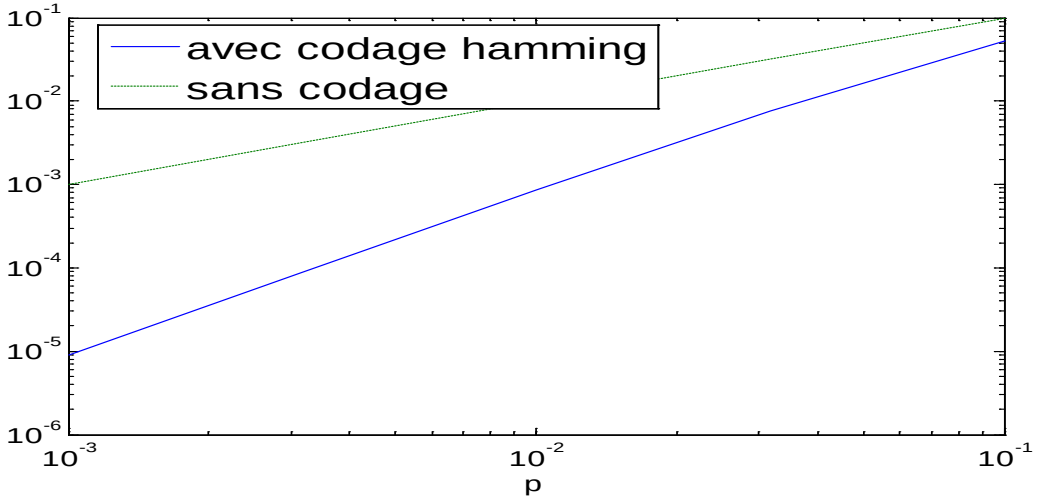
- Exercice: mettre H sous forme systématique et déterminer G
- Exercice: construire le tableau de décodage standard
- Exercice: décoder le mot 1100011

# Propriétés des codes de hamming

- Codes parfaits
  - tout mot de n bits s'écrit c+e avec c dans le code et e de poids au plus 1.
- Probabilité d'erreur mot: $$P = \sum_{i=2}^{n} C_n^i p^i (1-p)^{n-i}$$
- Probabilité d'erreur bit : (majoration)

$$P_{eb} \leq \frac{1}{n} \sum_{i=2}^{n} (i+1) C_n^i p^i (1-p)^{n-i}$$

  - 1er terme dominant si p petit
  - Exemple pour n=7



# Codage canal

- Généralités
- Théorie de l'information:
  - Information mutuelle
  - Capacité
  - Théorème du codage canal
- Codes en blocs
  - codage de Hamming
- **Codes cycliques**
- Codes convolutifs
  - algorithme de Viterbi

# Codes cycliques

- Codes linéaires qui ont la propriété suivante: toute permutation circulaire d'un mot de code donne un autre mot de code
- Représentation polynomiale:
  - $c = [c_{n-1} \dots c_1\ c_0]$ associé au polynôme $C(x) = c_{n-1} x^{n-1} + \dots + c_1 x + c_0$
    - Exemple: [ 1 1 0 1 ] $\rightarrow C(x) = x^3 + x^2 + 1$
- Permutation circulaire d'un bit à gauche: $xC(x)$ modulo $x^n+1$
  - $xC(x) = c_{n-1} x^n + c_{n-2} x^{n-1} + \dots + c_1 x^2 + c_0 x$
    $= c_{n-1} (x^n + 1) + c_{n-2} x^{n-1} + \dots + c_1 x^2 + c_0 x + c_{n-1}$
  - Reste de la division de $xC(x)$ par $x^n+1$ :
    $c_{n-2} x^{n-1} + \dots + c_0 x + c_{n-1}$ associé à $[c_{n-2} \dots c_0\ c_{n-1}]$
- Permutation de i bits à gauche : $x^iC(x)$ modulo $x^n+1$

# Polynôme générateur d'un code cyclique C(n,k)

- Polynôme générateur : g(x) de degré n-k qui divise $x^n+1$ :
  - $g(x) = x^{n-k} + g_{n-k-1} x^{n-k-1} + \ldots + g_1 x + g_0$
  - Exemple pour n=7 et k=4
    - $x^7 + 1 = (x+1)(x^3 + x^2 + 1)(x^3 + x + 1)$
    - 2 polynômes générateurs possibles
- Polynôme d'information correspondant au mot d'information $b = [b_{k-1} \ldots b_1 \; b_0]$ :
  - $B(x) = b_{k-1} x^{k-1} + \ldots + b_1 x + b_0$
- Polynôme du mot de code c associé à x: C(x)=g(x)B(x)
  - Exemple pour n=7 et k=4 avec $g(x) = (x^3 + x^2 + 1)$
    - $b = [0\ 0\ 1\ 1\ ] \rightarrow B(x)=x+1 \rightarrow C(x)=x^4 + x^2 + x + 1 \rightarrow c=[0\ 0\ 1\ 0\ 1\ 1\ 1]$
- On peut définir le code cyclique par ce seul polynôme générateur

# Matrice génératrice d'un code cyclique C(n,k)

- Les lignes de la matrice génératrice (non systématique) sont :
  - $x^{k-1}\, g(x)$
  - $x^{k-2}\, g(x)$
  - ...
  - $x\, g(x)$
  - $g(x)$
- Exemple pour n=7 et k=4 avec $g(x) = (x^3 + x^2 + 1)$
  - $$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

# Construction de codes systématiques

- Multiplier B(x) par $x^{n-k}$
  - Exemple (7,4), b = [0 0 1 1 ] $\leftrightarrow$ B(x)=x+1 : $(x+1)x^3 = x^4 + x^3$
- Diviser le résultat par g(x) : $B(x)\, x^{n-k} = Q(x)g(x)+r(x)$
  - Exemple avec $g(x) = (x^3 + x^2 + 1) : x^4 + x^3 = xg(x)+x$
- Ajouter le reste de la division r(x) à $B(x)x^{n-k}$ pour obtenir C(x) : $C(x) = B(x)\, x^{n-k} + r(x) = Q(x)g(x)$ qui est bien un mot de code (multiple de g(x)) et dont les bits de poids fort recopient le mot d'information
  - Exemple: $C(x)= x^4 + x^3 + x \leftrightarrow c = [0\ 0\ 1\ 1\ 0\ 1\ 0]$

# Codes cycliques dans les protocoles de communication – Cyclic Redundancy Check

- Problème: la taille des PDU (trame ou paquet) est variable (k et n non fixes)
- On fixe le nombre de bits de parité c'est à dire n-k
- On calcule ces bits de parité par la méthode précédente: reste de la division de B(x) $x^{n-k}$ par g(x)
- Exemples:
  - Trame HDLC (champ Frame Check Sequence (FCS) de 2 bytes)
    - $g(x) = x^{16} + x^{12} + x^5 + 1$ : 16 bits de parité
  - Trame Ethernet (champ CRC de 4 bytes)
    - $g(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+ x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
- Le CRC sert à détecter les erreurs:
  - Division du polynôme correspondant au mot reçu par g(x).
  - Si le reste de cette division est nul c'est qu'il n'y a pas d'erreur (ou que l'erreur est un mot de code)

# Codes BCH

- Codes cycliques construits pour assurer une distance minimale $d_{min} = 2t+1$
- Ils peuvent corriger t erreurs par mot de n bits
- Codes de longueur $n=2^m - 1$ avec $k \geq n - mt$
- Les polynômes générateurs sont construits en s'appuyant sur la théorie des corps de Galois
- On trouve des tables qui définissent ces polynômes pour les valeurs successives de n (voir slide suivant)
- Il existe des algorithmes de décodage de ces codes

# Table des codes BCH

- Notation des polynômes générateur en octal:
  - Exemple: $g(x) = x^4 + x + 1 \leftrightarrow [1\ 0\ 0\ 1\ 1] \leftrightarrow 23$ en octal

| n | k | t | g(x) |
|---|---|---|---|
| 7 | 4 | 1 | 13 |
| 15 | 11 | 1 | 23 |
| | 7 | 2 | 721 |
| | 5 | 3 | 2467 |
| 31 | 26 | 1 | 45 |
| | 21 | 2 | 3551 |
| ... | ... | | |
| 255 | 247 | 1 | 435561 |

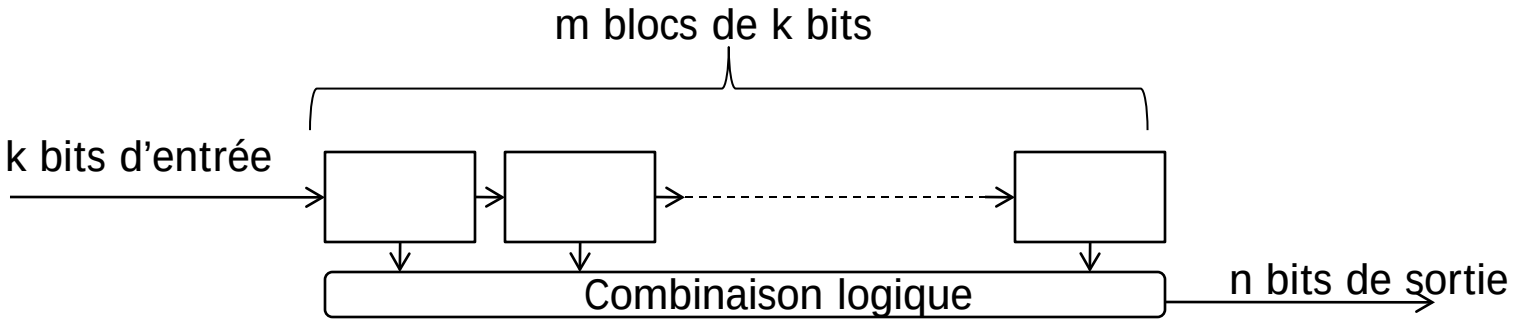
# Codage canal

- Généralités
- Théorie de l'information:
  - Information mutuelle
  - Capacité
  - Théorème du codage canal
- Codes en blocs
  - codage de Hamming
- Codes cycliques
- **Codes convolutifs**
  - algorithme de Viterbi

# Codes convolutifs

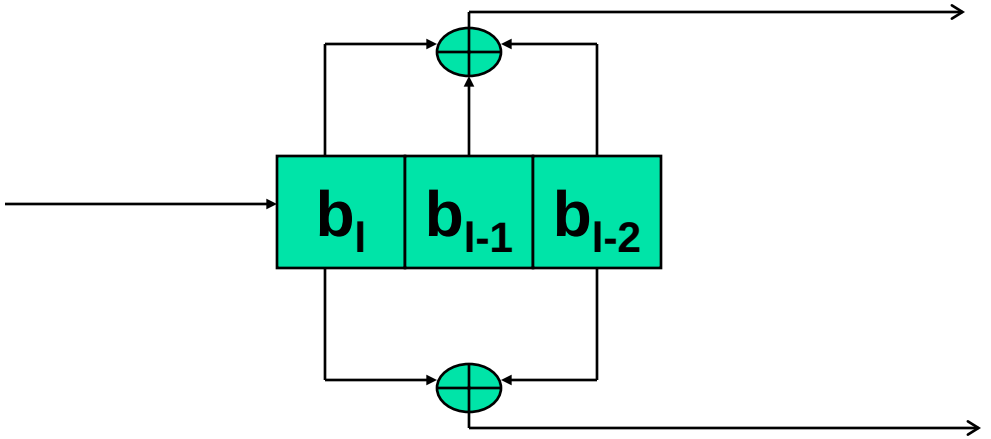
## Principe

- Chaque bloc de n bits de sortie dépend des m blocs de k bits d'information précédents (et non du seul bloc de k bits précédents).
- A chaque nouveau bloc de k bits d'entrée correspond un nouveau bloc de n bits de sortie.




- Caractéristiques:
  - Taux (rendement) : k/n
  - Longueur de contrainte : m
  - Les n « polynômes générateurs » $g^{(i)}$ (i=1:n) qui expriment les combinaisons des bits d'entrée pour obtenir les sorties
    - Dans le cas k=1: $c_l^{(i)} = \Sigma_{p=0:m-1}\ g_p^{(i)}\ b_{l-p}$ (l'expression a la forme d'une convolution)

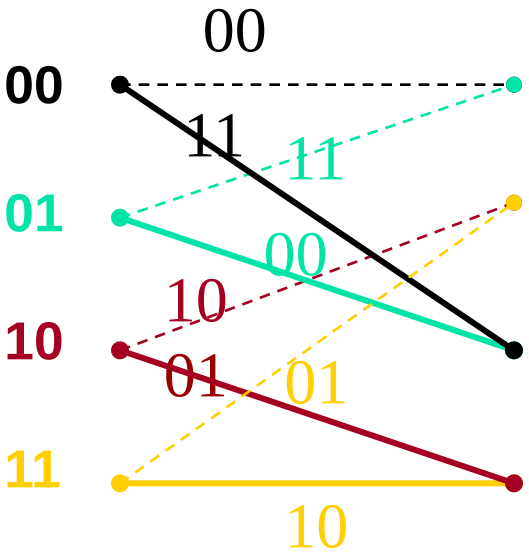
# Exemple de Code convolutif (n=2,k=1)




- Polynômes générateurs:
  - $g^{(1)}(x)=x^2+x+1$ soit $g^{(1)}=[1\ 1\ 1]$ soit 7 en octal
  - $g^{(2)}(x)=x^2+1$ soit $g^{(2)}=[1\ 0\ 1]$ soit 5 en octal
- Sa mémoire (m-1) ou longueur de contrainte (m)
  - Ici: m=3
- Exemple: codage de b=[1001] en partant de $b_{k-1}$ et $b_{k-2}=0$
  - $c^1(x) = g^{(1)}(x)\,(x^3+1) = x^5+x^4+x^3+x^2+x+1$
  - $c^2(x) = g^{(2)}(x)\,(x^3+1) = x^5+x^3+x^2+1$
  - c=[11 10 11 11 10 11] :les (m-1) dernières étapes correspondent à la remise à zéro du registre à décalage .

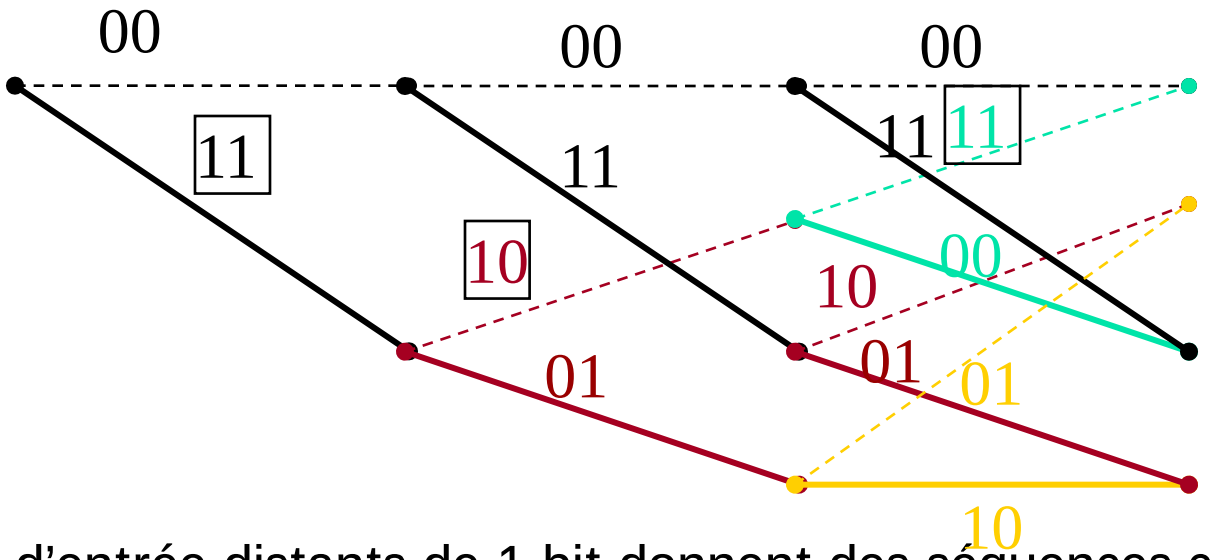
# Treillis d'un code convolutif (cas k=1)

- Les bits codés $c_l^i$ dépendent de $b_l$ et de l'état (contenu de la mémoire) à l'instant l : ($b_{l-1}$ $b_{l-2}$ … $b_{l-m+1}$).
- On peut représenter cette évolution par un treillis à $2^{m-1}$ états où :
  - les transitions sont notées par un trait différent suivant $b_l = 1$ ou $b_l = 0$
  - Les n bits de sortie figurent au dessus du trait de transition
- Exemple (g1=7,g2=5) :



# Distance libre

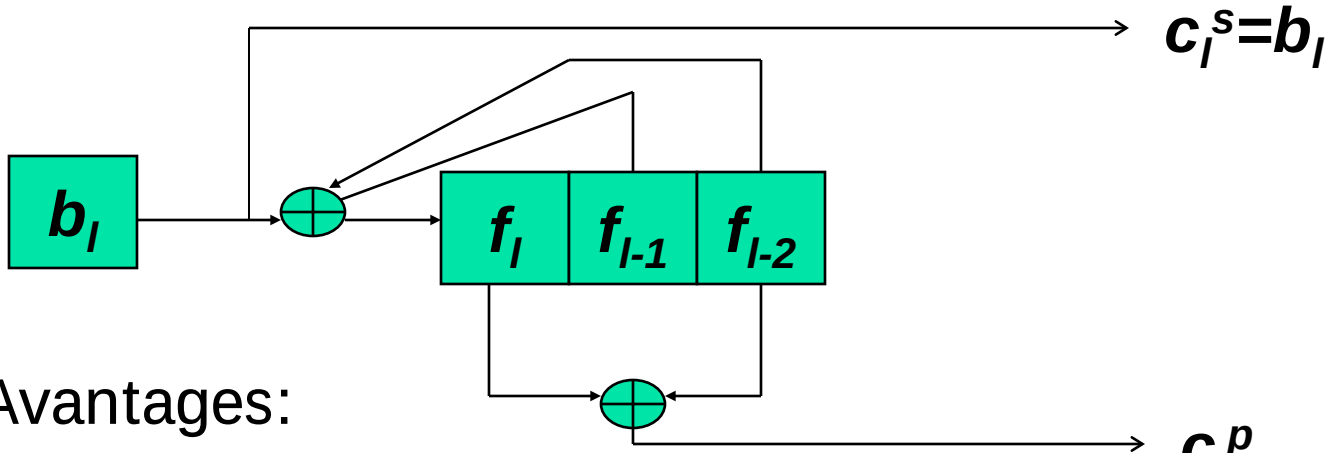
- C'est la distance minimale (au sens de hamming) entre deux chemins du treillis qui divergent et re-convergent.
  - Par linéarité c'est aussi le poids du chemin le plus court qui diverge du chemin tous-zéros et le rejoint.
  - Cette distance détermine les performances du code.
- Exemple (g1=7, g2=5) : $d_{libre}=5$ (w=1)




- Deux mots d'entrée distants de 1 bit donnent des séquences codées distantes de 5 bits

# code convolutif récursif systématique (RSC)

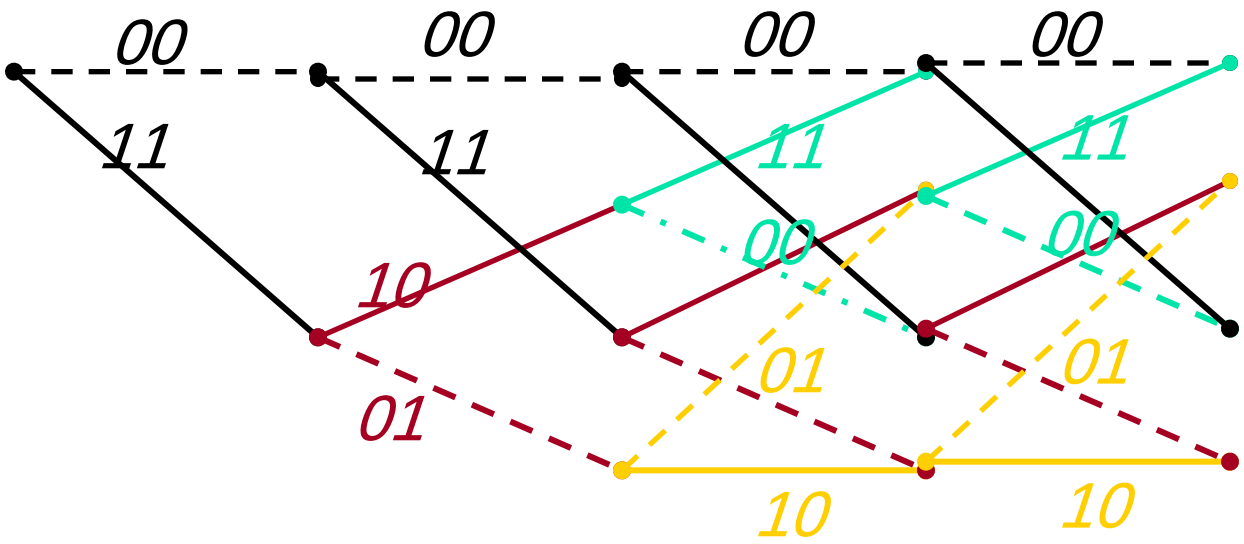
- La sortie dépend des sorties précédentes
- Les fonctions de transfert sont égales à 1 et $g^{(2)}(x)/g^{(1)}(x)$ dans le cas k=1, n=2.




- Avantages:
  - Bit systématique : identique à celui d'entrée
  - « Mots » d'entrée de poids 1 donnent des séquences codées de très grand poids
  - ⇔2 Mots d'entrée distants de 1 bit donnent des séquences codées très distantes
  - Ces codes RSC sont les codes constituants des turbo-codes usuels.

# Treillis de l'encodeur convolutif récursif systématique

- Le treillis d'un code RS est le même que celui du code NRNS associé (mais ne correspond pas aux mêmes bits entrants)
- exemple (g1=7,g2=5):
  - $d_{libre}$=5 obtenue pour mot de poids w = 3 ($1+x+x^2$)




  - Mot de poids w=2 ($1+x^3$) donne distance d=6

# Décodage des séquences codées au sens ML (maximum de vraisemblance): cas k=1,n=2

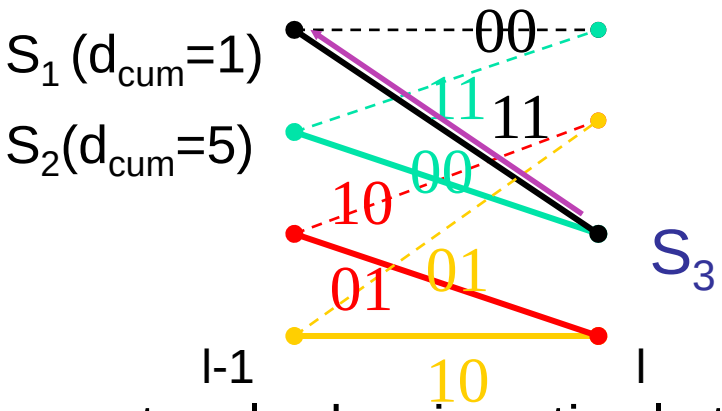
- Si on considère des séquences de bits d'information de L bits:
  - il y a $2^L$ séquences de bits possibles ($\Rightarrow$ $2^L$ chemins possibles dans le treillis de longueur L).
  - Les séquences codées sont de longueur 2L mais on note une séquence codée $c=[c_1\ c_2\ \ldots\ c_l\ \ldots\ c_L\ ]$ où $c_l=(c_l^1,c_l^2)$
- La séquence reçue est de la forme: r = c + w où w est un bruit additif
- On recherche de la séquence $b=[b_1,b_2,\ldots,b_L]$ qui maximise la probabilité d'observer la séquence reçue $r=[r_1,r_2,\ldots,r_L]$ .
- Si le bruit est blanc $p(r/b^{(p)})=\prod p(w_l=r_l-c_l^{(p)})$
  - S'il est gaussien: il faut trouver la séquence de bits $b^{(p)}$ qui minimise: $D_{cum}=\sum_l \left| r_l-c_l^{(p)} \right|^2$ où $\{c_l^{(p)}\}$ est la séquence en sortie du codeur pour l'entrée $b^{(p)}$
  - La recherche exhaustive est très complexe puisqu'il y a $2^L$ séquences possibles et pour chacune, le calcul de D nécessite celui de L distances.

# Algorithme de Viterbi

- Cet algorithme permet d'avoir une complexité en L* $2^{(m-1)}$ au lieu de L*$2^{L}$
- Algorithme de Viterbi:
  - à chaque instant l, pour chaque état $S_i$ , on garde le meilleur des $2^k$ chemins qui aboutissent à cet état.

$$d_{cum}(l, S_i) = \min_{S_j}\left(d_{cum}(l-1, S_j) + d\left(r_l, c(S_j, S_i)\right)\right)$$

  - c($S_j$ ,$S_i$ ) est la séquence de 2 bits codés correspondant à la transition $S_j \rightarrow S_i$
  - d($r_l$ , c($S_j$ ,$S_i$ )) est la distance locale à l'étape l qui peut être prise comme distance euclidienne (critère ML) ou distance de hamming.
  - Exemple

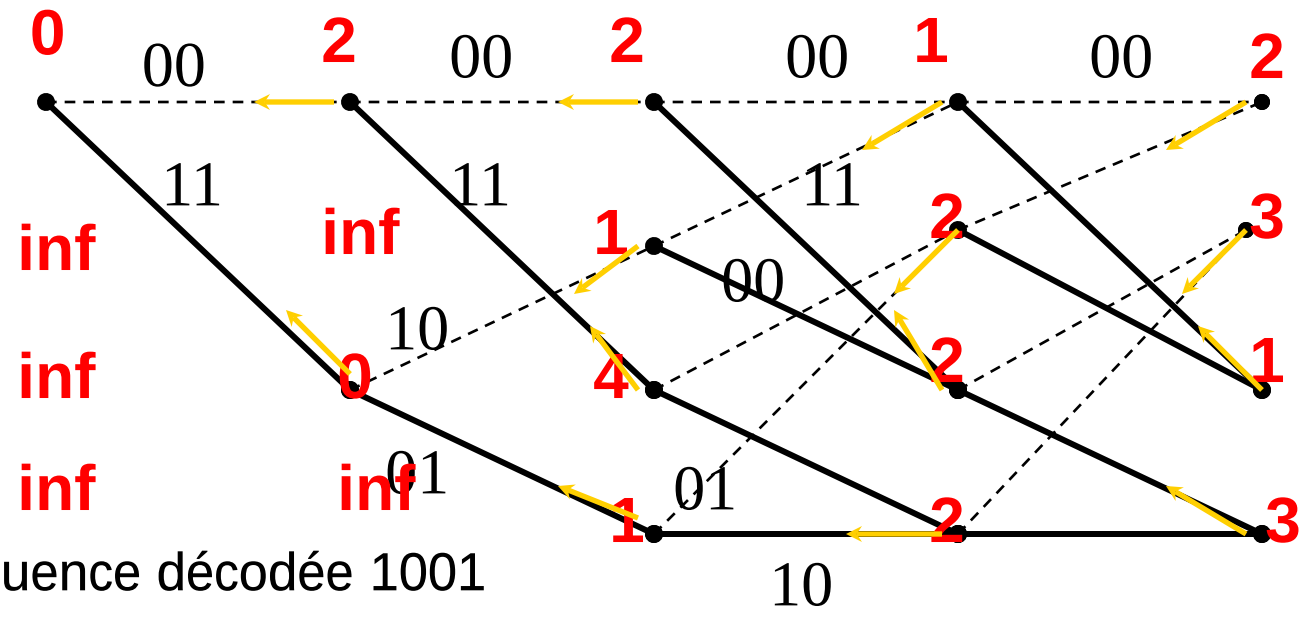



Exemple: signal reçu $y_l$=00
$d_{cum}(l,S_3)$=min(1+2,5+0)=3

- A la fin on « remonte » le chemin optimal et on le traduit en séquence émise

# Exemple pour Viterbi ($d_{cum}$)

- Séquence émise 11 10 11 11
- Séquence reçue 11 00 11 11




- Séquence décodée 1001