

TS345
-
Codage pour la 5G

Romain Tajan

19 octobre 2026

TS345 en bref...

Organisation du module

- 6 créneaux (1h20) de cours
- 3 créneaux de TP (2h40)

Découpage des cours

- 1 créneau de **rappels sur les codes correcteurs**
- 3 créneaux sur les **Codes LDPC**
- 2 créneaux sur les **Codes Polaires**

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Plan

- 1 Introduction générale
 - ▷ Histoire de code correcteur
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Plan

- 1 Introduction générale
 - ▷ Histoire de code correcteur
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Un peu d'histoire...

- 
- 1948 **Shannon** - capacité d'un canal (non constructive)
 - 1955 **Elias** - Code convolutifs (GSM)
 - 1960 **Reed et Solomon** - Codes RS (CD → BluRay, QR, DVB-S, RAID6)
Gallager - Codes LDPC
 - 1966 **Forney** - Codes concatennés (Pioneer (1968-1972), Voyager (1977))
 - 1967 **Viterbi** - Décodage optimal des codes convolutifs
 - 1993 **Berrou, Glavieux et Thitimajshima** - Turbocodes (3G/4G, deep-space)
 - 1996 **MacKay** - Ré-invente les LDPC (DVB-S2, WiFi, 5G)
 - 2008 **Arikan** - Codes Polaires (5G)

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
 - ▷ Sur la modélisation du canal
 - ▷ Code correcteur d'erreur
 - ▷ Probabilité d'erreur
 - ▷ Retour sur les enjeux
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
 - ▷ Sur la modélisation du canal
 - ▷ Code correcteur d'erreur
 - ▷ Probabilité d'erreur
 - ▷ Retour sur les enjeux
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Le canal...

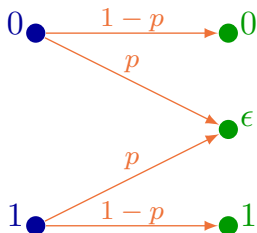
Un **canal** est défini par un triplet : $(\mathcal{X}, \mathcal{Y}, p(y|x))$ où

- $\mathcal{X}$ est l'**alphabet d'entrée**
- $\mathcal{Y}$ est l'**alphabet de sortie**
- $p(y|x)$ est la **probabilité de transition**

Soit $n \in \mathbb{N}$ et soit le canal $(\mathcal{X}^n, \mathcal{Y}^n, p(\mathbf{y}|\mathbf{x}))$, ce canal est dit "**sans mémoire**" si sa probabilité de transition vérifie

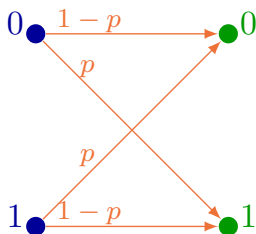
$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^n p(y_i|x_i)$$

Le canal à effacement binaire



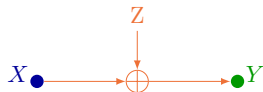
- $\mathcal{X} = \{0, 1\}$ (canal à entrées binaires)
- $\mathcal{Y} = \{0, \epsilon, 1\}$
- $p(\epsilon|0) = p(\epsilon|1) = p$ et $p(0|0) = p(1|1) = 1 - p$
- Canal utile pour les couches hautes, pour le stockage

Le canal binaire symétrique



- $\mathcal{X} = \{0, 1\}$ (canal à entrées binaires)
- $\mathcal{Y} = \{0, 1\}$
- $p(1|0) = p(0|1) = p$ et $p(0|0) = p(1|1) = 1 - p$
- Canal utile après décision

Le canal additif gaussien

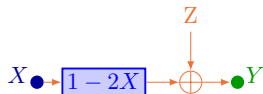


- $\mathcal{X} = \mathbb{R}$

- $\mathcal{Y} = \mathbb{R}$

- $p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y-x)^2}$

Le canal additif gaussien à entrées binaires



- $\mathcal{X} = \{0, 1\}$

- $\mathcal{Y} = \mathbb{R}$

- $p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y-1+2x)^2}$

Plan

- 1 Introduction générale
- 2 **Rappels sur de codage / définitions**
 - ▷ Sur la modélisation du canal
 - ▷ **Code correcteur d'erreur**
 - ▷ Probabilité d'erreur
 - ▷ Retour sur les enjeux
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Code (M, n)

Un code (M, n) pour le canal $(\mathcal{X}^n, \mathcal{Y}^n, p(\mathbf{y}|\mathbf{x}))$ est composé de 3 éléments

- Un ensemble de M **messages**. On notera cet ensemble $\mathcal{M} = \{0, 1, \dots, M-1\}$
- Une fonction d'**encodage** (ou encodeur) notée ϕ :

$$\begin{aligned}\phi : \mathcal{M} &\rightarrow \mathcal{X}^n \\ W &\mapsto \mathbf{X} = \phi(W)\end{aligned}$$

$\phi(\cdot)$ doit être **injective**

- Une fonction de **décodage** (ou décodeur) notée ψ :

$$\begin{aligned}\psi : \mathcal{Y}^n &\rightarrow \mathcal{M} \\ \mathbf{Y} &\mapsto \hat{W} = \psi(\mathbf{Y})\end{aligned}$$

$\psi(\cdot)$ doit être **surjective**

Plan

- 1 Introduction générale
- 2 **Rappels sur de codage / définitions**
 - ▷ Sur la modélisation du canal
 - ▷ Code correcteur d'erreur
 - ▷ **Probabilité d'erreur**
 - ▷ Retour sur les enjeux
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Probabilité d'erreur

Si le mot de code $W = w$ est envoyé, une erreur se produit ssi $\hat{W} \neq w$.

La probabilité associée à cet événement est notée

$$\begin{aligned}\lambda_w &= \mathbb{P}(\hat{W} \neq w | W = w) \\ &= \mathbb{P}(\psi(\mathbf{Y}) \neq w | W = w)\end{aligned}$$

Définitions

- **Probabilité d'erreur maximale** : $P_m^{(n)} = \max_w \lambda_w$
- **Probabilité d'erreur moyenne** : $P_e^{(n)} = \mathbb{P}(\hat{W} \neq W) = \frac{1}{M} \sum_{w=0}^{M-1} \lambda_w$

Décodage du Maximum a Posteriori (MAP)

Définition

- Soit $\mathcal{C}$ un code (M, n) donné.
- Le **décodeur du Maximum A Posteriori (MAP)** est la fonction de $\mathbf{y}$ définie par :

$$\Psi_{MAP}(\mathbf{y}) = \operatorname{argmax}_{w \in \mathcal{M}} \mathbb{P}(W = w | \mathbf{Y} = \mathbf{y})$$

Le décodeur MAP minimise P_e

Décodage MAP sur canaux classiques

Soit le **décodeur** MAP défini par : $\Psi_{MAP}(\mathbf{y}) = \underset{w \in \mathcal{M}}{\operatorname{argmax}} \mathbb{P}(W = w | \mathbf{Y} = \mathbf{y})$

- 1 Sur canal BSC : $\Psi_{MAP}(\mathbf{y}) = \underset{\mathbf{x} \in \mathcal{C}}{\operatorname{argmin}} d_H(\mathbf{x}, \mathbf{y})$
- 2 Sur canal AWGN : $\Psi_{MAP}(\mathbf{y}) = \underset{\mathbf{x} \in \mathcal{C}}{\operatorname{argmin}} d_E(\mathbf{x}, \mathbf{y})$

Sans structure sur $\mathcal{C}$, ces deux décodeurs sont trop complexes !

Décodage du Maximum a Posteriori (MAP-bit)

Définition

- Soit $\mathcal{C}$ un code **binaire** (k, n) donné.
- Le **décodeur du Maximum A Posteriori bit (MAP-bit)** est la fonction de $\mathbf{y}$ définie par :

$$\psi_{MAP-bit}^{(j)}(\mathbf{y}) = \operatorname{argmax}_{u \in \{0,1\}} \mathbb{P}(U_j = u | \mathbf{Y} = \mathbf{y})$$

- En pratique on calcule les **Logarithmes de rapports de vraisemblances** (LLR) :

$$L(U_i) = \log \frac{\mathbb{P}(U_i = 0 | \mathbf{y})}{\mathbb{P}(U_i = 1 | \mathbf{y})}$$

- Le décodeur MAP minimise P_b (la probabilité d'erreur binaire)
- Le signe des LLRs : décisions MAP-bit
- Le module des LLRs : fiabilité des décisions

Plan

- 1 Introduction générale
- 2 **Rappels sur de codage / définitions**
 - ▷ Sur la modélisation du canal
 - ▷ Code correcteur d'erreur
 - ▷ Probabilité d'erreur
 - ▷ **Retour sur les enjeux**
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Enjeux du codage

Compromis entre

- La **taille** du code (n)
- Le **rendement de code** (le débit)
- La **probabilité d'erreur** (maximale ou moyenne)
- La **complexité** de l'encodage
- La **complexité** du décodage

Efficacité spectrale $\iff$ Codage $\iff$ Efficacité énergétique

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
 - ▷ Matrice de parité
 - ▷ Encodeur Systématique
 - ▷ Décodage MAP-bit des codes linéaires (binaires)
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Avant de commencer...

Remarques

- ➊ Dans cette section $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ et le canal considéré est le **canal binaire symétrique**
- ➋ Dans cette section on notera $\mathbb{F}_2$ le **corps** $(\{0, 1\}, \oplus, \cdot)$ où :
 - Pour $x, y \in \mathbb{F}_2$, $x \oplus y = (x + y) \bmod 2$ ($\equiv$ OU exclusif)
 - Pour $x, y \in \mathbb{F}_2$, $x \cdot y$ est le produit "classique" entre x et y ($\equiv$ ET)
- ➌ $\mathbb{F}_2$ est un corps fini à deux éléments $(\mathbb{Z}/2\mathbb{Z})$
- ➍ Par la suite on notera $\oplus \rightsquigarrow +$
- ➎ $(\mathbb{F}_2^n, +, \cdot)$ est un **espace vectoriel** où
 - Pour $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, $\mathbf{x} + \mathbf{y} = [x_0 + y_0, x_1 + y_1, \dots, x_{n-1} + y_{n-1}]$
 - Pour $x \in \mathbb{F}_2$ et $\mathbf{y} \in \mathbb{F}_2^n$, $x \cdot \mathbf{y} = [x \cdot y_0, x \cdot y_1, \dots, x \cdot y_{n-1}]$

Code linéaire en bloc

Code linéaire

Soit $\mathcal{C}$ un code ($M = 2^k, n$), $\mathcal{C}$ est un **code binaire linéaire** si et seulement si les mots de codes $\mathbf{c} \in \mathbb{F}_2^n$ sont obtenus à partir des messages $\mathbf{u} \in \mathbb{F}_2^k$ par la relation

$$\mathbf{c} = \mathbf{u}G$$

où G est une matrice de taille $k \times n$ appelée **matrice génératrice** de $\mathcal{C}$

$$G = \begin{pmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{pmatrix} = \begin{pmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{pmatrix}$$

Remarques

- ① $\mathcal{C}$ est un sous-espace vectoriel de $\mathbb{F}_2^n$ de dimension $\text{rang}(G) = k$
- ② Il existe plusieurs matrices génératrices pour un même code.
- ③ le rendement du code est $R = \frac{\text{rang}(G)}{n} = \frac{k}{n}$

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
 - ▷ Matrice de parité
 - ▷ Encodeur Systématique
 - ▷ Décodage MAP-bit des codes linéaires (binaires)
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Code dual | Matrice de parité

Matrice de parité

Le code $\mathcal{C}$ peut aussi être défini par sa **matrice de parité** H de taille $n - k \times n$:

$$H = \begin{pmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{n-k-1} \end{pmatrix} = \begin{pmatrix} h_{0,0} & h_{0,1} & \dots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \dots & h_{1,n-1} \\ \vdots & \vdots & & \vdots \\ h_{n-k-1,0} & h_{n-k-1,1} & \dots & h_{n-k-1,n-1} \end{pmatrix}$$

Soit $\mathbf{v} \in \mathbb{F}_2^n$, $\mathbf{v} \in \mathcal{C}$ ($\mathbf{v}$ est un mot de code) si et seulement si

$$\mathbf{v}H^T = 0$$

- 1 H est appelée **matrice de parité** du code $\mathcal{C}$ et vérifie $GH^T = 0_{k \times n-k}$
- 2 H n'est pas unique

Codes linéaires en blocs

Définitions

- 1 À partir de sa **matrice génératrice** G de taille $k \times n$: $\mathcal{C} = \{\mathbf{u}G \mid \mathbf{u} \in \mathbb{F}_2^k\}$
- 2 À partir de sa **matrice de parité** H de taille $n - k \times n$: $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n \mid \mathbf{c}H^T = \mathbf{0}\}$

- 1 G et H **ne sont pas uniques**
- 2 G et H vérifient $GH^T = 0_{k \times n-k}$. Vrai pour tout couple de matrices (G, H) définissant un même code
- 3 Pour un code binaire : $k \leq n \Rightarrow$ le codage "**ajoute de la redondance**"
- 4 **Rendement de code** :

$$R = \frac{\text{rang}(G)}{n} = \frac{n - \text{rang}(H)}{n}$$

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
 - ▷ Matrice de parité
 - ▷ Encodeur Systématique
 - ▷ Décodage MAP-bit des codes linéaires (binaires)
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Encodeur systématique

Soit $\mathcal{C}$ un code ($M = 2^k, n$) pour un canal à entrées binaires. Un encodeur $\varphi(\cdot)$ est dit **systématique** ssi

$$\forall \mathbf{u} \in \mathbb{F}_2^k, \varphi(\mathbf{u}) = [\mathbf{p} \ \mathbf{u}] \text{ avec } \mathbf{p} \in \mathbb{F}_2^{n-k}$$

Si $\mathcal{C}$ est linéaire alors il existe une matrice génératrice sous la forme

$$G = \begin{pmatrix} p_{0,0} & \dots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & \dots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{k,0} & \dots & p_{k,n-k-1} & 0 & 0 & \dots & 1 \end{pmatrix} = [P \ I_k]$$

La matrice de parité associée à la matrice G précédente

$$H = \begin{pmatrix} 1 & 0 & \dots & 0 & p_{0,0} & \dots & p_{k,0} \\ 0 & 1 & \dots & 0 & p_{0,1} & \dots & p_{k,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & p_{0,n-k-1} & \dots & p_{k,n-k-1} \end{pmatrix} = [I_{n-k} \ P^T]$$

Remarques sur les encodeurs systématiques

$$G = \begin{pmatrix} p_{0,0} & \dots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & \dots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{k,0} & \dots & p_{k,n-k-1} & 0 & 0 & \dots & 1 \end{pmatrix} = [P \quad I_k]$$

- 1 Un encodeur systématique comporte le message **en clair**

Remarques sur les encodeurs systématiques

$$G = \begin{pmatrix} p_{0,0} & \dots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & \dots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{k,0} & \dots & p_{k,n-k-1} & 0 & 0 & \dots & 1 \end{pmatrix} = [P \quad I_k]$$

- 1 Un encodeur systématique comporte le message **en clair**
- 2 Les encodeurs systématiques sont souvent moins complexes que leurs équivalents non-systématiques

Remarques sur les encodeurs systématiques

$$G = \begin{pmatrix} p_{0,0} & \dots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & \dots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{k,0} & \dots & p_{k,n-k-1} & 0 & 0 & \dots & 1 \end{pmatrix} = [P \ I_k]$$

- 1 Un encodeur systématique comporte le message **en clair**
- 2 Les encodeurs systématiques sont souvent moins complexes que leurs équivalents non-systématiques
- 3 Une matrice d'encodage systématique peut être trouvée pour tout code linéaire en bloc de matrice génératrice **pleine** (à des permutations de colonnes près)
 $\rightsquigarrow$ **Pivot de Gauss**

Exemple de Pivot de Gauss

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

- 1 But : permuter | sommer des lignes pour faire apparaître la matrice I à droite
- 2 Cette procédure ne donne pas tout le temps une matrice de la forme $G = [P, I]$
- 3 Si G est de rang plein on peut toujours se ramener à $[P, I]$ à **une permutation de colonne près**
- 4 Soit $G' = [P, I_k] = G\Pi$ où Π est une matrice de permutation des colonnes, soit $H' = [I_{n-k} P^T]$ alors

$$G'(H')^T = 0_{k \times n-k} = GH^T \text{ avec } H = H'\Pi^T$$

Exemple de Pivot de Gauss

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

- ❶ But : permuter | sommer des lignes pour faire apparaître la matrice I à droite
- ❷ Cette procédure ne donne pas tout le temps une matrice de la forme $G = [P, I]$
- ❸ Si G est de rang plein on peut toujours se ramener à $[P, I]$ à une permutation de colonne près
- ❹ Soit $G' = [P, I_k] = G\Pi$ où Π est une matrice de permutation des colonnes, soit $H' = [I_{n-k} P^T]$ alors

$$G'(H')^T = 0_{k \times n-k} = GH^T \text{ avec } H = H'\Pi^T$$

Exemple de Pivot de Gauss

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{array}{l} \leftarrow \text{Pivot} \\ L_2 \leftarrow L_2 + L_1 \end{array}$$

- ❶ But : permuter | sommer des lignes pour faire apparaître la matrice I à droite
- ❷ Cette procédure ne donne pas tout le temps une matrice de la forme $G = [P, I]$
- ❸ Si G est de rang plein on peut toujours se ramener à $[P, I]$ **à une permutation de colonne près**
- ❹ Soit $G' = [P, I_k] = G\Pi$ où Π est une matrice de permutation des colonnes, soit $H' = [I_{n-k} P^T]$ alors

$$G'(H')^T = 0_{k \times n-k} = GH^T \text{ avec } H = H'\Pi^T$$

Exemple de Pivot de Gauss

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

- ❶ But : permuter | sommer des lignes pour faire apparaître la matrice I à droite
- ❷ Cette procédure ne donne pas tout le temps une matrice de la forme $G = [P, I]$
- ❸ Si G est de rang plein on peut toujours se ramener à $[P, I]$ à une permutation de colonne près
- ❹ Soit $G' = [P, I_k] = G\Pi$ où Π est une matrice de permutation des colonnes, soit $H' = [I_{n-k} P^T]$ alors

$$G'(H')^T = 0_{k \times n-k} = GH^T \text{ avec } H = H'\Pi^T$$

Exemple de Pivot de Gauss

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{array}{l} \leftarrow \text{Pivot} \\ L_3 \leftarrow L_3 + L_2 \end{array}$$

- ❶ But : permuter | sommer des lignes pour faire apparaître la matrice I à droite
- ❷ Cette procédure ne donne pas tout le temps une matrice de la forme $G = [P, I]$
- ❸ Si G est de rang plein on peut toujours se ramener à $[P, I]$ **à une permutation de colonne près**
- ❹ Soit $G' = [P, I_k] = G\Pi$ où Π est une matrice de permutation des colonnes, soit $H' = [I_{n-k} P^T]$ alors

$$G'(H')^T = 0_{k \times n-k} = GH^T \text{ avec } H = H'\Pi^T$$

Exemple de Pivot de Gauss

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

- ❶ But : permuter | sommer des lignes pour faire apparaître la matrice I à droite
- ❷ Cette procédure ne donne pas tout le temps une matrice de la forme $G = [P, I]$
- ❸ Si G est de rang plein on peut toujours se ramener à $[P, I]$ à une permutation de colonne près
- ❹ Soit $G' = [P, I_k] = G\Pi$ où Π est une matrice de permutation des colonnes, soit $H' = [I_{n-k} P^T]$ alors

$$G'(H')^T = 0_{k \times n-k} = GH^T \text{ avec } H = H'\Pi^T$$

Exemple de Pivot de Gauss

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{array}{l} \leftarrow \text{Pivot} \\ L_4 \leftarrow L_4 + L_3 \end{array}$$

- ❶ But : permuter | sommer des lignes pour faire apparaître la matrice I à droite
- ❷ Cette procédure ne donne pas tout le temps une matrice de la forme $G = [P, I]$
- ❸ Si G est de rang plein on peut toujours se ramener à $[P, I]$ **à une permutation de colonne près**
- ❹ Soit $G' = [P, I_k] = G\Pi$ où Π est une matrice de permutation des colonnes, soit $H' = [I_{n-k} P^T]$ alors

$$G'(H')^T = 0_{k \times n-k} = GH^T \text{ avec } H = H'\Pi^T$$

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
 - ▷ Matrice de parité
 - ▷ Encodeur Systématique
 - ▷ Décodage MAP-bit des codes linéaires (binaires)
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations

Décodage MAP-bit

Décodage MAP-bit

- Le **décodeur MAP-bit** encodage systématique :

$$\Psi_{MAP-bit}^{(j)}(\mathbf{y}) = \operatorname{argmax}_{x_j \in \{0,1\}} \mathbb{P}(X_j = x_j | \mathbf{Y} = \mathbf{y})$$

Décodage MAP-bit

Décodage MAP-bit

- Le **décodeur MAP-bit** encodage systématique :

$$\Psi_{MAP-bit}^{(j)}(\mathbf{y}) = \operatorname{argmax}_{x_j \in \{0,1\}} \mathbb{P}(X_j = x_j | \mathbf{Y} = \mathbf{y})$$

- Le **décodeur MAP-bit** encodage systématique (2) :

$$\Psi_{MAP-bit}^{(j)}(\mathbf{y}) = \operatorname{argmax}_{x_j \in \{0,1\}} \sum_{\mathbf{x} \sim_j \in \mathbb{F}_2^{n-1}} \mathbb{P}(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) \mathbb{1}(\mathbf{x}H^T = \mathbf{0})$$

Décodage MAP-bit

Décodage MAP-bit

- Le **décodeur MAP-bit** encodage systématique :

$$\Psi_{MAP-bit}^{(j)}(\mathbf{y}) = \operatorname{argmax}_{x_j \in \{0,1\}} \mathbb{P}(X_j = x_j | \mathbf{Y} = \mathbf{y})$$

- Le **décodeur MAP-bit** encodage systématique (2) :

$$\begin{aligned} \Psi_{MAP-bit}^{(j)}(\mathbf{y}) &= \operatorname{argmax}_{x_j \in \{0,1\}} \sum_{\mathbf{x} \sim_j \in \mathbb{F}_2^{n-1}} \mathbb{P}(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) \mathbb{1}(\mathbf{x}H^T = \mathbf{0}) \\ &= \operatorname{argmax}_{x_j \in \{0,1\}} \sum_{\mathbf{x} \sim_j \in \mathbb{F}_2^{n-1}} \prod_{i=0}^{n-1} \mathbb{P}(Y_i = y_i | X_i = x_i) \mathbb{1}(\mathbf{x}H^T = \mathbf{0}) \end{aligned}$$

Décodage MAP-bit

Décodage MAP-bit

- Le **décodeur MAP-bit** encodage systématique :

$$\Psi_{MAP-bit}^{(j)}(\mathbf{y}) = \operatorname{argmax}_{x_j \in \{0,1\}} \mathbb{P}(X_j = x_j | \mathbf{Y} = \mathbf{y})$$

- Le **décodeur MAP-bit** encodage systématique (2) :

$$\begin{aligned} \Psi_{MAP-bit}^{(j)}(\mathbf{y}) &= \operatorname{argmax}_{x_j \in \{0,1\}} \sum_{\mathbf{x} \sim_j \in \mathbb{F}_2^{n-1}} \mathbb{P}(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) \mathbb{1}(\mathbf{x}H^T = \mathbf{0}) \\ &= \operatorname{argmax}_{x_j \in \{0,1\}} \sum_{\mathbf{x} \sim_j \in \mathbb{F}_2^{n-1}} \prod_{i=0}^{n-1} \mathbb{P}(Y_i = y_i | X_i = x_i) \mathbb{1}(\mathbf{x}H^T = \mathbf{0}) \end{aligned}$$

Sans structure sur $\mathcal{C}$, ce décodeur est aussi trop complexe !

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 **LDPC**
 - ▷ Présentation générale
 - ▷ Définition
 - ▷ Graphe de Tanner associé à un code LDPC
 - ▷ Décodage Somme-Produit
 - ▷ Étude des performances du décodage itératif
- 5 Évolution de densité

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 **LDPC**
 - ▷ Présentation générale
 - ▷ Définition
 - ▷ Graphe de Tanner associé à un code LDPC
 - ▷ Décodage Somme-Produit
 - ▷ Étude des performances du décodage itératif
- 5 Évolution de densité

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)
 - Codes représentable à l'aide d'un **graphe bipartite** (**graphe de Tanner**)

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)
 - Codes représentable à l'aide d'un **graphe bipartite** (**graphe de Tanner**)
 - Décodage possible à l'aide du graphe

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)
 - Codes représentable à l'aide d'un **graphe bipartite** (**graphe de Tanner**)
 - Décodage possible à l'aide du graphe
 - Performances dépendant des propriétés du graphe

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)
 - Codes représentable à l'aide d'un **graphe bipartite** (**graphe de Tanner**)
 - Décodage possible à l'aide du graphe
 - Performances dépendant des propriétés du graphe
- Algorithme de propagation de croyance (IA) (Pearl en 1988)

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)
 - Codes représentable à l'aide d'un **graphe bipartite (graphe de Tanner)**
 - Décodage possible à l'aide du graphe
 - Performances dépendant des propriétés du graphe
- Algorithme de propagation de croyance (IA) (Pearl en 1988)
 - Algorithme de propagation de croyance (BP - Belief Propagation)

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)
 - Codes représentable à l'aide d'un **graphe bipartite (graphe de Tanner)**
 - Décodage possible à l'aide du graphe
 - Performances dépendant des propriétés du graphe
- Algorithme de propagation de croyance (IA) (Pearl en 1988)
 - Algorithme de propagation de croyance (BP - Belief Propagation)
- **Redécouverte** des codes LDPC (MacKay, Luby fin 1990)

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)
 - Codes représentable à l'aide d'un **graphe bipartite (graphe de Tanner)**
 - Décodage possible à l'aide du graphe
 - Performances dépendant des propriétés du graphe
- Algorithme de propagation de croyance (IA) (Pearl en 1988)
 - Algorithme de propagation de croyance (BP - Belief Propagation)
- **Redécouverte** des codes LDPC (MacKay, Luby fin 1990)
 - (Re)Montrent que les codes LDPC sont de bons codes

Codes Low Density Parity Check (LDPC)

- Introduits par Gallager pendant sa **thèse de doctorat** en 1963
 - Codes possédant une **matrice de parité peu dense**
 - Codes pouvant être analysés (exposant d'erreur)
 - Décodage simplifié
- **Peu de travaux** pendant ~ 30 ans (Tanner en 1981)
 - Codes représentable à l'aide d'un **graphe bipartite (graphe de Tanner)**
 - Décodage possible à l'aide du graphe
 - Performances dépendant des propriétés du graphe
- Algorithme de propagation de croyance (IA) (Pearl en 1988)
 - Algorithme de propagation de croyance (BP - Belief Propagation)
- **Redécouverte** des codes LDPC (MacKay, Luby fin 1990)
 - (Re)Montrent que les codes LDPC sont de bons codes

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 **LDPC**
 - ▷ Présentation générale
 - ▷ **Définition**
 - ▷ Graphe de Tanner associé à un code LDPC
 - ▷ Décodage Somme-Produit
 - ▷ Étude des performances du décodage itératif
- 5 Évolution de densité

Définition des codes LDPC

Définitions

- 1 Soit une matrice H

$$H = \begin{pmatrix} h_{0,0} & h_{0,1} & \dots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \dots & h_{1,n-1} \\ \vdots & \vdots & & \vdots \\ h_{m-1,0} & h_{m-1,1} & \dots & h_{m-1,n-1} \end{pmatrix}$$

Densité de H : $\frac{|\{i,j : h_{i,j} = 1\}|}{m n}$

- 2 **Codes LDPC** : Codes possédant une matrice de parité H peu dense (creuse). Ordre de grandeur pour n grand ≤ 0.01 .
- 3 **Codes réguliers** : poids des lignes constant r , poids des colonnes constant g
- 4 Rendement d'un code LDPC régulier : $R \geq 1 - \frac{m}{n} = 1 - \frac{g}{r}$
- 5 $R_d = 1 - \frac{g}{r}$ est appelé **rendement de construction** d'un code LDPC

Définition des codes LDPC

Petit TD dans le cours...

Soit une matrice H

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

- 1 Donner la densité de H
- 2 H définit-elle un code LDPC régulier ?
- 3 Si oui, que valent g et r ?
- 4 Combien vaut le rendement de construction de ce code ?
- 5 Combien vaut le rendement de ce code ?

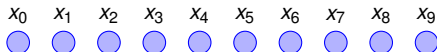
Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 **LDPC**
 - ▷ Présentation générale
 - ▷ Définition
 - ▷ **Graphe de Tanner associé à un code LDPC**
 - ▷ Décodage Somme-Produit
 - ▷ Étude des performances du décodage itératif
- 5 Évolution de densité

Graphe de Tanner

Le **graphe de Tanner** est un **graphe bipartite** avec :

- 1 n **nœuds de variables** représentant les variables v_j $j \in \{0, \dots, n-1\}$
- 2 m **nœuds de parité** p_i $i \in \{0, \dots, m-1\}$
- 3 Une arête est dessinée entre nœud de variable x_j et le nœud de parité c_i ssi $h_{i,j} = 1$

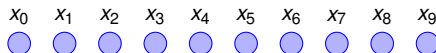


$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

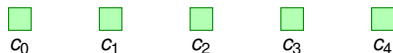
Graphe de Tanner

Le **graphe de Tanner** est un **graphe bipartite** avec :

- 1 n **nœuds de variables** représentant les variables $v_j, j \in \{0, \dots, n-1\}$
- 2 m **nœuds de parité** $p_i, i \in \{0, \dots, m-1\}$
- 3 Une arête est dessinée entre nœud de variable x_j et le nœud de parité c_i ssi $h_{i,j} = 1$



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

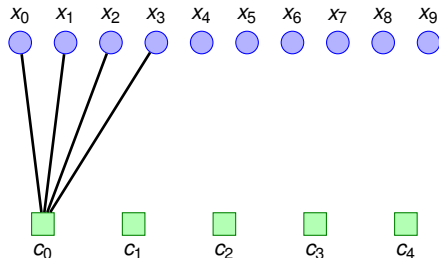


Graphe de Tanner

Le **graphe de Tanner** est un **graphe bipartite** avec :

- 1 n **nœuds de variables** représentant les variables $v_j, j \in \{0, \dots, n-1\}$
- 2 m **nœuds de parité** $p_i, i \in \{0, \dots, m-1\}$
- 3 Une arête est dessinée entre nœud de variable x_j et le nœud de parité c_i ssi $h_{i,j} = 1$

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

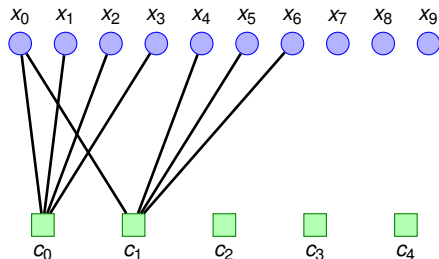


Graphe de Tanner

Le **graphe de Tanner** est un **graphe bipartite** avec :

- 1 n **nœuds de variables** représentant les variables $v_j, j \in \{0, \dots, n-1\}$
- 2 m **nœuds de parité** $p_i, i \in \{0, \dots, m-1\}$
- 3 Une arête est dessinée entre nœud de variable x_j et le nœud de parité c_i ssi $h_{i,j} = 1$

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

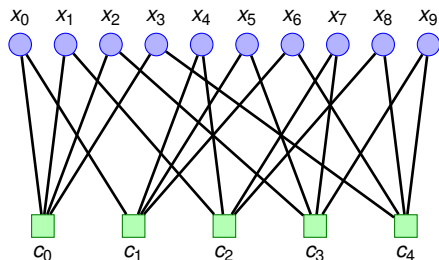


Graphe de Tanner

Le **graphe de Tanner** est un **graphe bipartite** avec :

- 1 n **nœuds de variables** représentant les variables $v_j, j \in \{0, \dots, n-1\}$
- 2 m **nœuds de parité** $p_i, i \in \{0, \dots, m-1\}$
- 3 Une arête est dessinée entre nœud de variable x_j et le nœud de parité c_i ssi $h_{i,j} = 1$

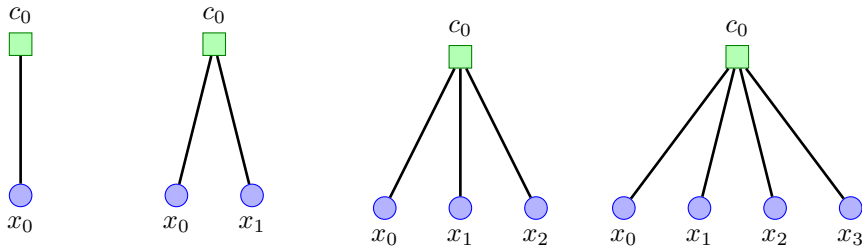
$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$



Degrés des nœuds de variable

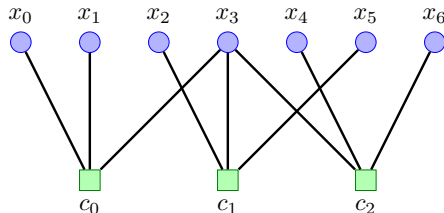
 x_0  c_0 x_0  c_0 c_1 x_0  c_0 c_1 c_2 x_0  c_0 c_1 c_2 c_3

Degrés des nœuds de parité



Codes LDPC irréguliers

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



Polynôme de distribution des degrés des nœuds de variables : $\lambda(X) = \sum_{d=1}^{d_v} \lambda_d X^{d-1}$

Polynôme de distribution des degrés des nœuds de parités : $\rho(X) = \sum_{d=1}^{d_c} \rho_d X^{d-1}$

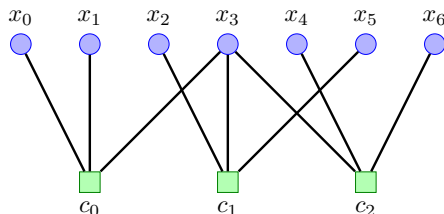
Borne sur le rendement du code : $R \geq 1 - \frac{\int_0^1 \rho(x) dx}{\int_0^1 \lambda(x) dx}$

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 **LDPC**
 - ▷ Présentation générale
 - ▷ Définition
 - ▷ Graphe de Tanner associé à un code LDPC
 - ▷ **Décodage Somme-Produit**
 - ▷ Étude des performances du décodage itératif
- 5 Évolution de densité

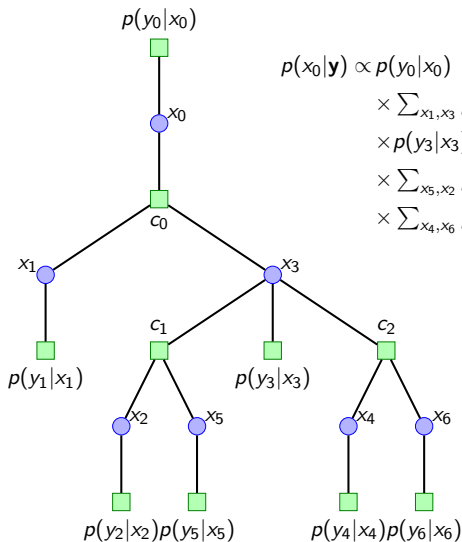
Retour sur le décodage du MAP-bit

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



$$\begin{aligned} p(x_0|\mathbf{y}) &\propto \sum_{\mathbf{x} \sim 0} \prod_{i=0}^6 p(y_i|x_i) \mathbb{1}(\mathbf{x}H^T = \mathbf{0}) \\ &= p(y_0|x_0) \sum_{x_1, x_3} p(y_1|x_1)p(y_3|x_3) \mathbb{1}(x_0 + x_1 + x_3 = 0) \\ &\quad \times \sum_{x_5, x_2} p(y_2|x_2)p(y_5|x_5) \mathbb{1}(x_2 + x_3 + x_5 = 0) \\ &\quad \times \sum_{x_4, x_6} p(y_4|x_4)p(y_6|x_6) \mathbb{1}(x_3 + x_4 + x_6 = 0) \end{aligned}$$

Algorithme somme-produit



$$p(x_0|y) \propto p(y_0|x_0)$$

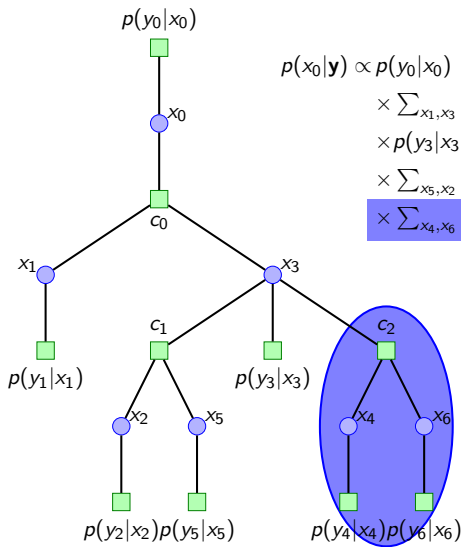
$$\times \sum_{x_1, x_3} p(y_1|x_1) \mathbb{1}(x_0 + x_1 + x_3 = 0)$$

$$\times p(y_3|x_3)$$

$$\times \sum_{x_5, x_2} p(y_2|x_2) p(y_5|x_5) \mathbb{1}(x_2 + x_3 + x_5 = 0)$$

$$\times \sum_{x_4, x_6} p(y_4|x_4) p(y_6|x_6) \mathbb{1}(x_3 + x_4 + x_6 = 0)$$

Algorithme somme-produit



$$p(x_0|y) \propto p(y_0|x_0)$$

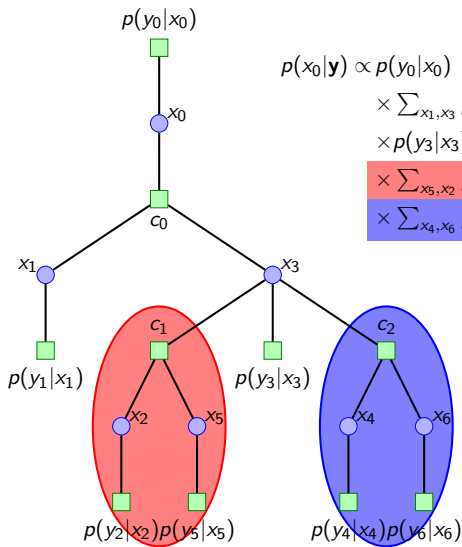
$$\times \sum_{x_1, x_3} p(y_1|x_1) \mathbb{1}(x_0 + x_1 + x_3 = 0)$$

$$\times p(y_3|x_3)$$

$$\times \sum_{x_5, x_2} p(y_2|x_2) p(y_5|x_5) \mathbb{1}(x_2 + x_3 + x_5 = 0)$$

$$\times \sum_{x_4, x_6} p(y_4|x_4) p(y_6|x_6) \mathbb{1}(x_3 + x_4 + x_6 = 0)$$

Algorithme somme-produit



$$p(x_0|\mathbf{y}) \propto p(y_0|x_0)$$

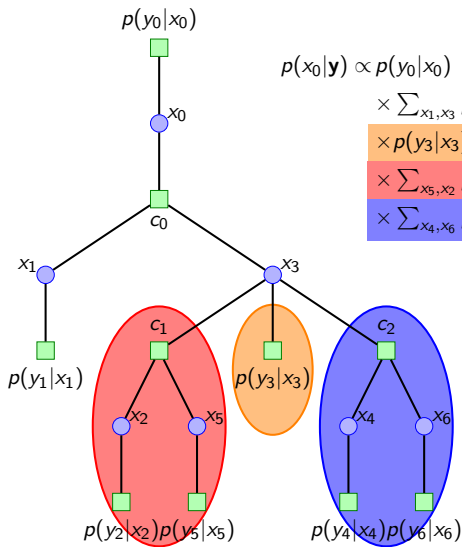
$$\times \sum_{x_1, x_3} p(y_1|x_1) \mathbb{1}(x_0 + x_1 + x_3 = 0)$$

$$\times p(y_3|x_3)$$

$$\times \sum_{x_5, x_2} p(y_2|x_2) p(y_5|x_5) \mathbb{1}(x_2 + x_3 + x_5 = 0)$$

$$\times \sum_{x_4, x_6} p(y_4|x_4) p(y_6|x_6) \mathbb{1}(x_3 + x_4 + x_6 = 0)$$

Algorithme somme-produit



$$p(x_0|y) \propto p(y_0|x_0)$$

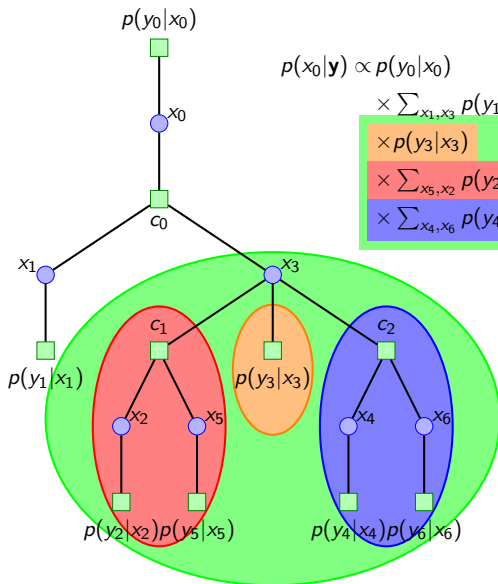
$$\times \sum_{x_1, x_3} p(y_1|x_1) \mathbb{1}(x_0 + x_1 + x_3 = 0)$$

$$\times p(y_3|x_3)$$

$$\times \sum_{x_5, x_2} p(y_2|x_2) p(y_5|x_5) \mathbb{1}(x_2 + x_3 + x_5 = 0)$$

$$\times \sum_{x_4, x_6} p(y_4|x_4) p(y_6|x_6) \mathbb{1}(x_3 + x_4 + x_6 = 0)$$

Algorithme somme-produit



$$p(x_0|y) \propto p(y_0|x_0)$$

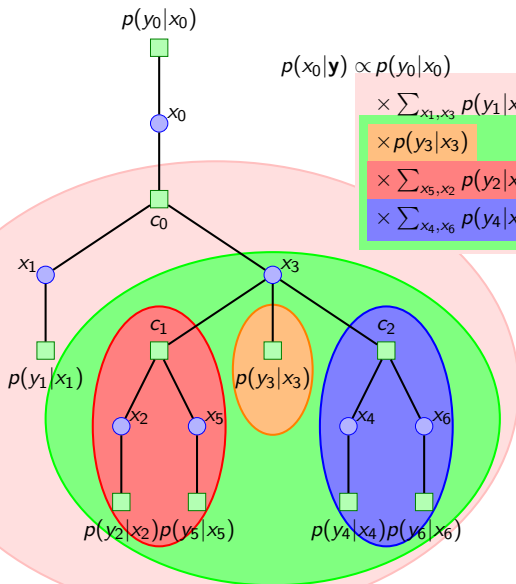
$$\times \sum_{x_1, x_3} p(y_1|x_1) \mathbb{1}(x_0 + x_1 + x_3 = 0)$$

$$\times p(y_3|x_3)$$

$$\times \sum_{x_5, x_2} p(y_2|x_2) p(y_5|x_5) \mathbb{1}(x_2 + x_3 + x_5 = 0)$$

$$\times \sum_{x_4, x_6} p(y_4|x_4) p(y_6|x_6) \mathbb{1}(x_3 + x_4 + x_6 = 0)$$

Algorithme somme-produit



$$p(x_0|y) \propto p(y_0|x_0)$$

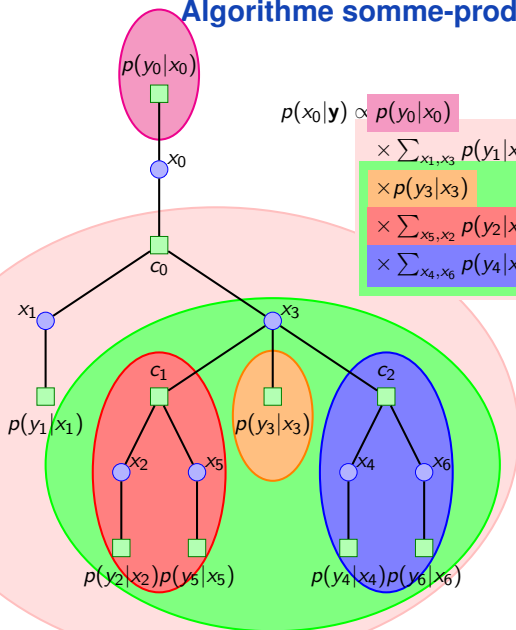
$$\times \sum_{x_1, x_3} p(y_1|x_1) \mathbb{1}(x_0 + x_1 + x_3 = 0)$$

$$\times p(y_3|x_3)$$

$$\times \sum_{x_5, x_2} p(y_2|x_2) p(y_5|x_5) \mathbb{1}(x_2 + x_3 + x_5 = 0)$$

$$\times \sum_{x_4, x_6} p(y_4|x_4) p(y_6|x_6) \mathbb{1}(x_3 + x_4 + x_6 = 0)$$

Algorithme somme-produit



$$p(x_0|y) \propto p(y_0|x_0)$$

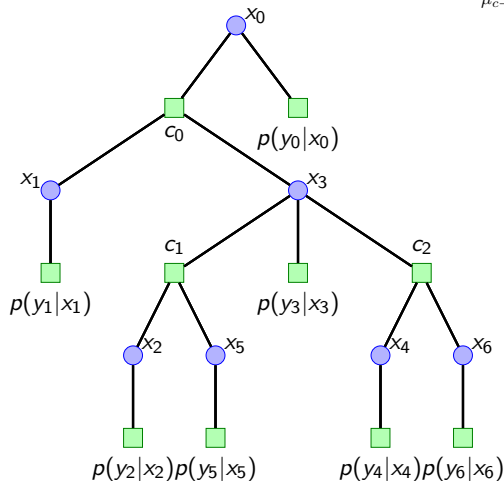
$$\times \sum_{x_1, x_3} p(y_1|x_1) \mathbb{1}(x_0 + x_1 + x_3 = 0)$$

$$\times p(y_3|x_3)$$

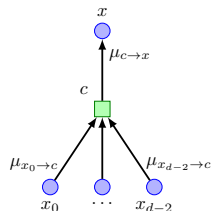
$$\times \sum_{x_5, x_2} p(y_2|x_2) p(y_5|x_5) \mathbb{1}(x_2 + x_3 + x_5 = 0)$$

$$\times \sum_{x_4, x_6} p(y_4|x_4) p(y_6|x_6) \mathbb{1}(x_3 + x_4 + x_6 = 0)$$

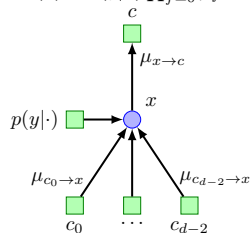
Algorithme somme-produit



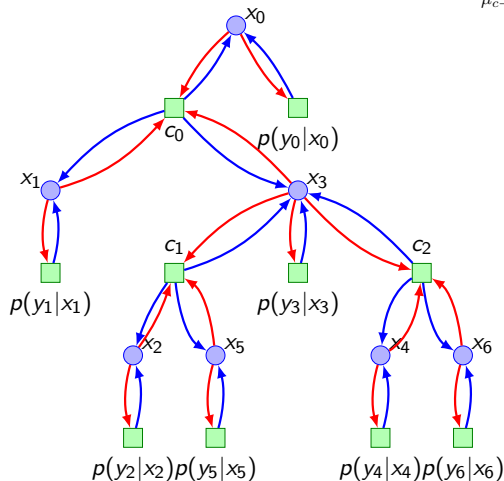
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



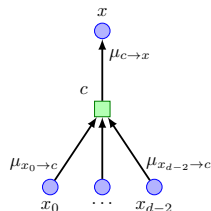
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



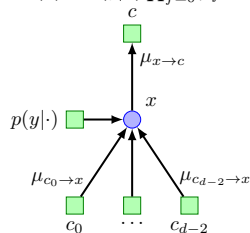
Algorithme somme-produit



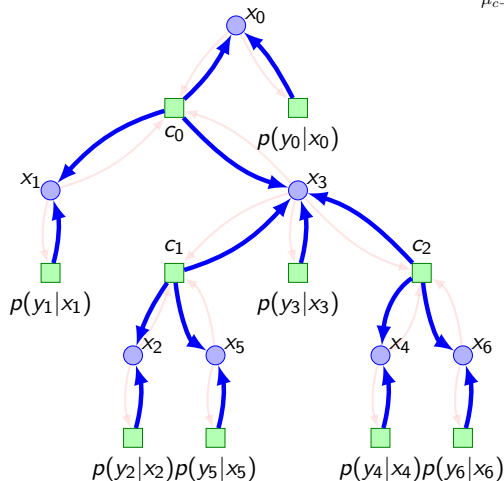
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_0^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



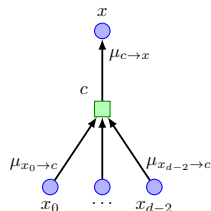
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



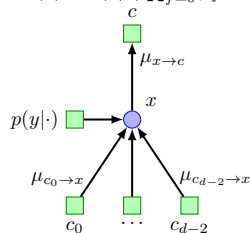
Algorithme somme-produit



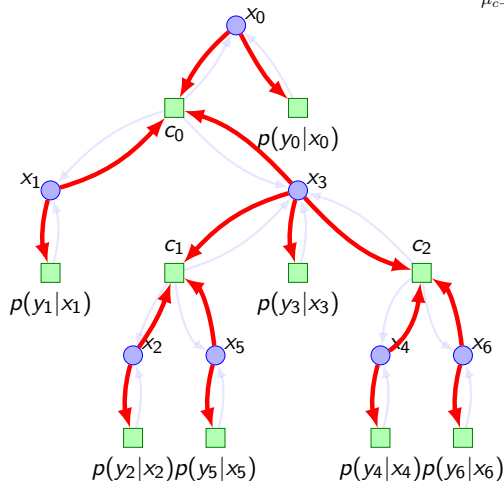
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_0^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



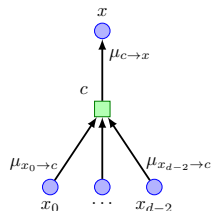
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



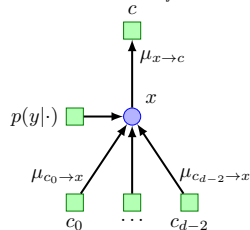
Algorithme somme-produit



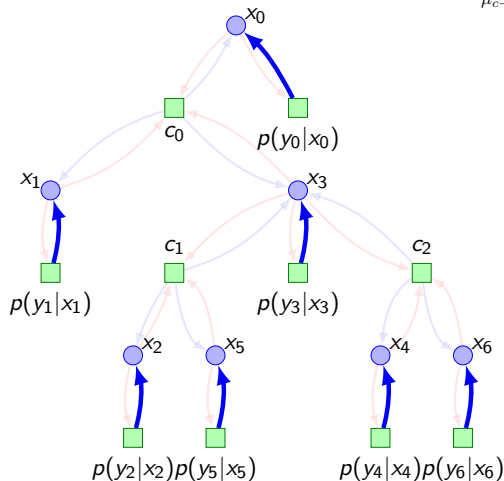
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_0^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



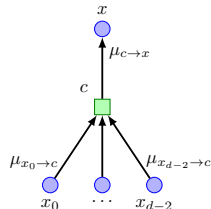
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



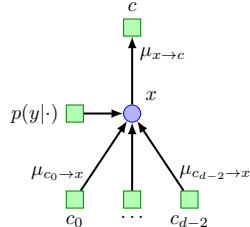
Algorithme somme-produit



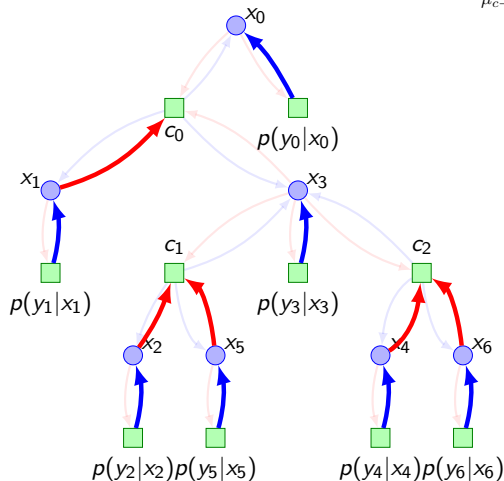
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



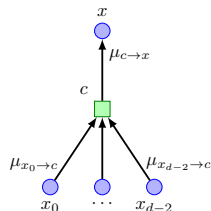
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



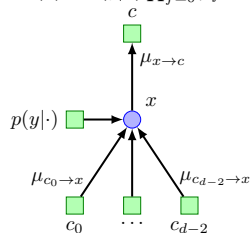
Algorithme somme-produit



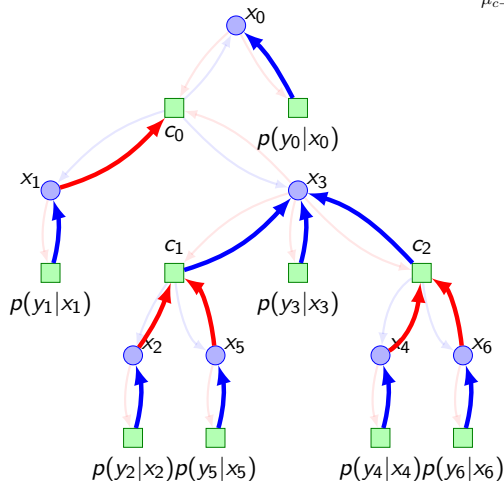
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



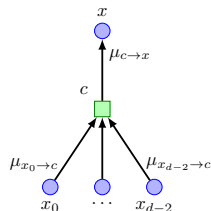
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



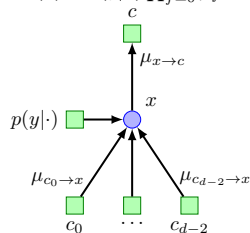
Algorithme somme-produit



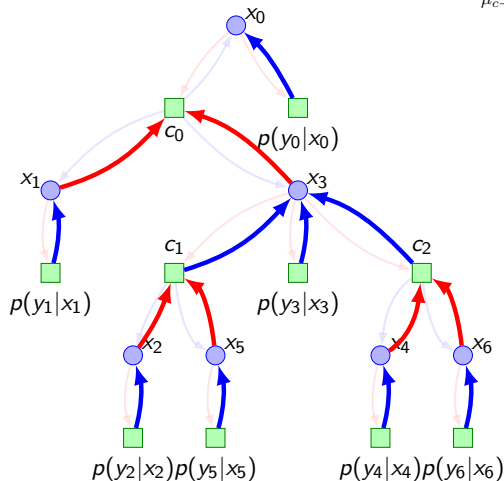
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



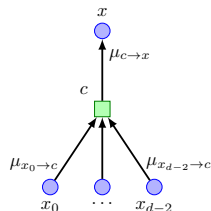
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



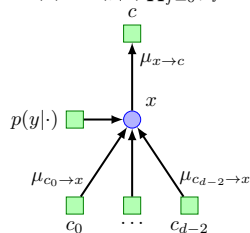
Algorithme somme-produit



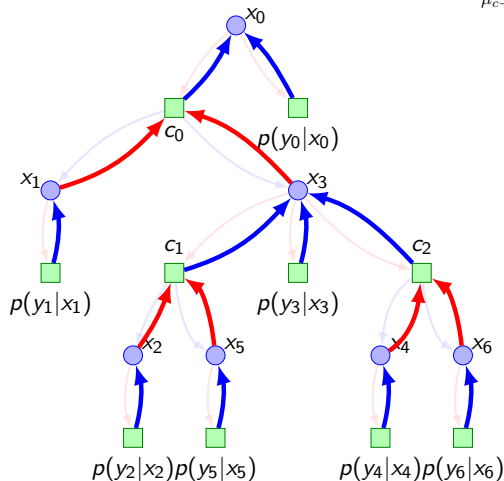
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



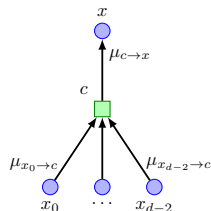
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



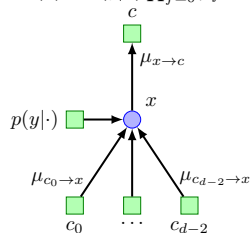
Algorithme somme-produit



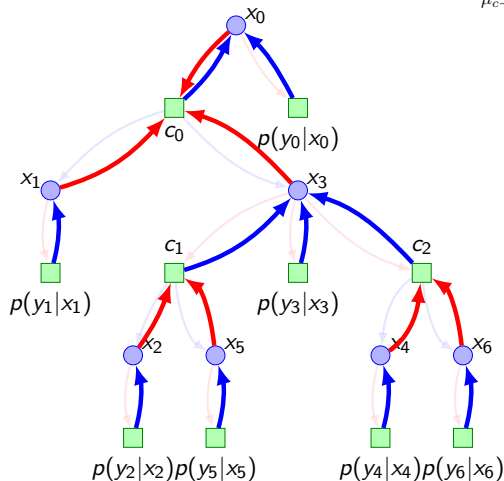
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



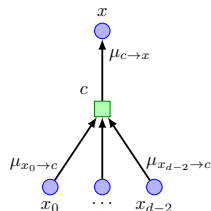
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



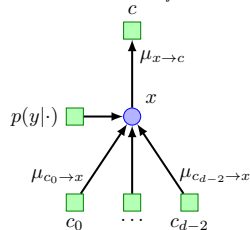
Algorithme somme-produit



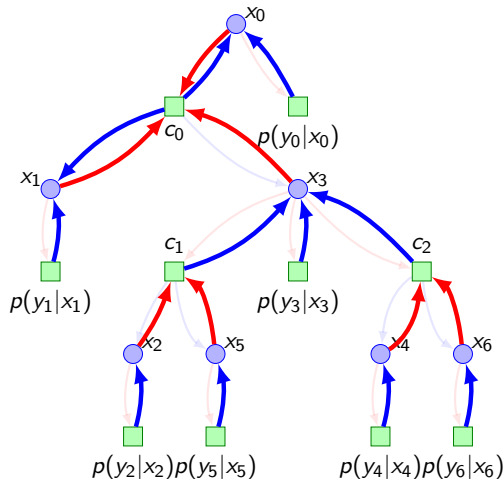
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



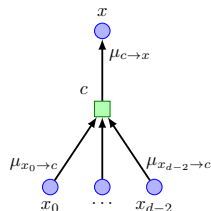
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



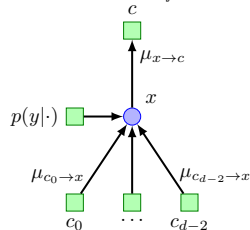
Algorithme somme-produit



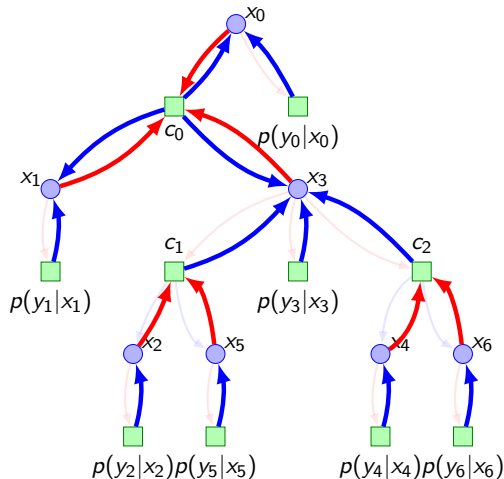
$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$



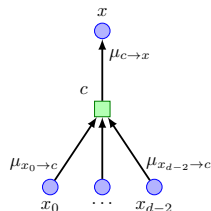
$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



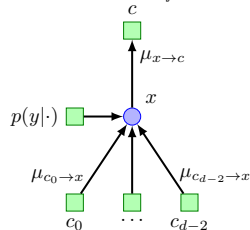
Algorithme somme-produit



$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$

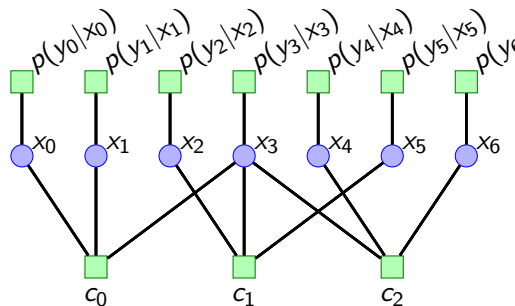


$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$



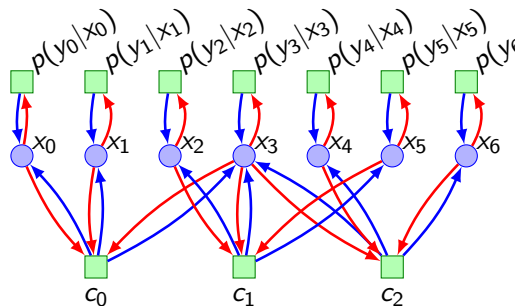
Ordonnancement : flooding

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



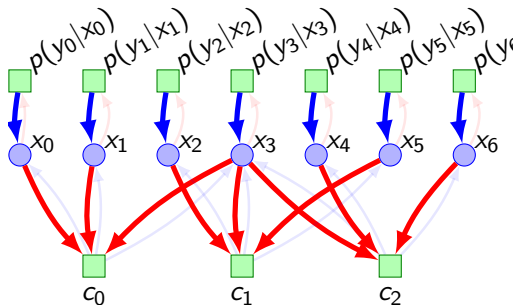
Ordonnancement : flooding

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



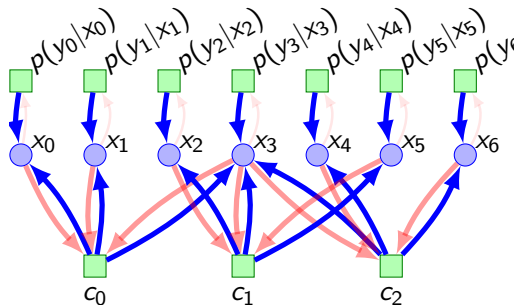
Ordonnancement : flooding

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



Ordonnancement : flooding

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



Conclusion sur l'algorithme somme-produit

L'algorithme SP :

- utilise des calcul locaux $\Rightarrow$ réduit complexité

Conclusion sur l'algorithme somme-produit

L'algorithme SP :

- utilise des calcul locaux $\Rightarrow$ réduit complexité
- permet d'approximer les probabilités a posteriori $p(x_i|\mathbf{y})$

Conclusion sur l'algorithme somme-produit

L'algorithme SP :

- utilise des calcul locaux $\Rightarrow$ réduit complexité
- permet d'approximer les probabilités a posteriori $p(x_i|\mathbf{y})$
- ce calcul est **exact** si le **graphe de Tanner est un arbre**

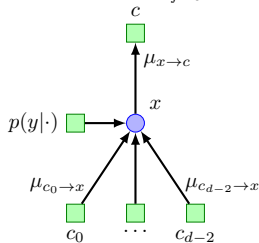
Conclusion sur l'algorithme somme-produit

L'algorithme SP :

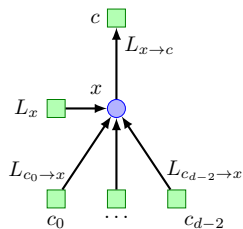
- utilise des calcul locaux $\Rightarrow$ réduit complexité
- permet d'approximer les probabilités a posteriori $p(x_i|\mathbf{y})$
- ce calcul est **exact** si le **graphe de Tanner est un arbre**
- si le graphe possède des cycles $\Rightarrow$ itérer

Calculer avec des LLR - Nœuds de variables

$$\mu_{x \rightarrow c}(x) = p(y|x) \prod_{f=0}^{d-2} \mu_{f \rightarrow x}(x)$$

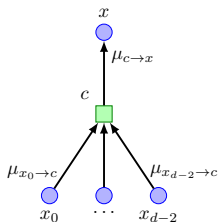


$$L_{x \rightarrow c} = L_x + \sum_{k=0}^{d-2} L_{c_k \rightarrow x}$$

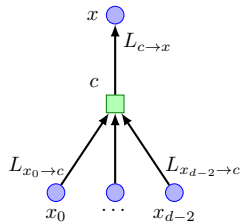

 $\Rightarrow$

Calculer avec des LLR - Nœuds de variables

$$\mu_{c \rightarrow x}(x) = \sum_{\sim x} \mathbb{1}(x + x_0 + \dots + x_{d-2} = 0) \times \prod_{k=0}^{d-2} \mu_{x_k \rightarrow c}(x_k)$$


 $\Rightarrow$

$$L_{c \rightarrow x} = 2 \operatorname{atanh} \left(\prod_{k=0}^{d-2} \tanh \frac{L_{x_k \rightarrow c}}{2} \right)$$



Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 **LDPC**
 - ▷ Présentation générale
 - ▷ Définition
 - ▷ Graphe de Tanner associé à un code LDPC
 - ▷ Décodage Somme-Produit
 - ▷ Étude des performances du décodage itératif
- 5 Évolution de densité

Propriétés

- **Théorème de concentration** : Les performances des codes longs d'un même ensemble (avec les mêmes $\lambda(X)$ et $\rho(X)$) se comportent globalement de la même manière $\Rightarrow$ on peut réaliser une étude en moyenne.
- **Pour des codes longs**, étude moyenne $\Leftrightarrow$ étude des codes acycliques ayant les mêmes $\lambda(X)$ et $\rho(X)$
- **Pour des codes longs**, effet de seuil sur le **paramètre du canal** (probabilité d'erreur, probabilité d'effacement ou SNR)

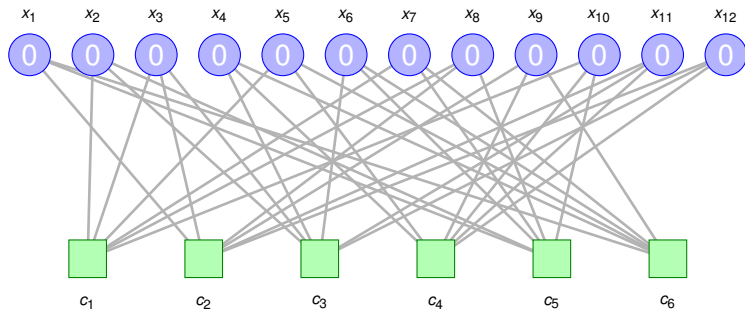
Code LDPC et matrice de parité

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

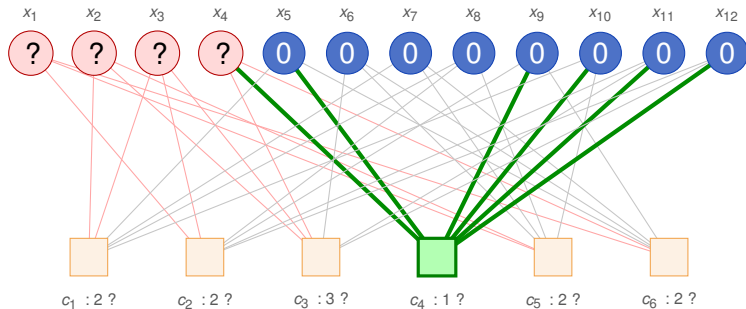
$$\mathcal{C} = \{\mathbf{x} \in \mathbb{F}_2^{12} : H\mathbf{x}^\top = \mathbf{0}\}$$

- **Peu dense** : 3 uns par colonne, 6 par ligne.
- Rendement $R \geq 1 - \frac{3}{6} = \frac{1}{2}$.
- Distributions d'arêtes :
 $\lambda(x) = x^2$, $\rho(x) = x^5$.

Le graphe de Tanner

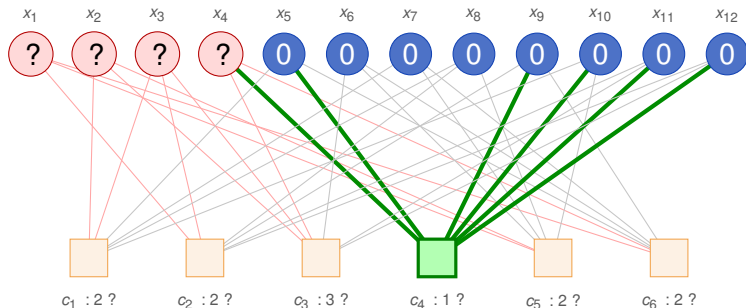


Exemple : 4 effacements, décodage réussi



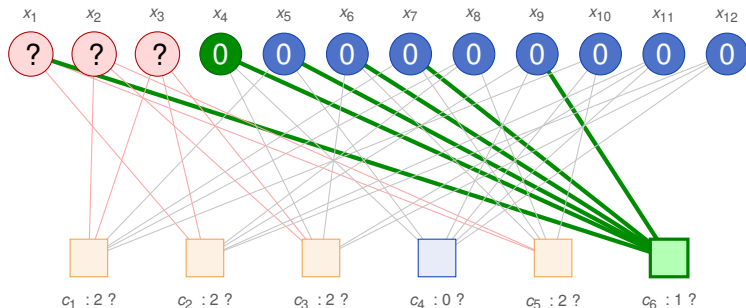
Mot reçu : x_1, x_2, x_3, x_4 effacés (fraction $4/12 \approx 0,33$). Chaque contrôle indique le nombre e de voisins effacés : $e=1 \Rightarrow$ résolvable, $e \geq 2 \Rightarrow$ bloqué.

Exemple : 4 effacements, décodage réussi



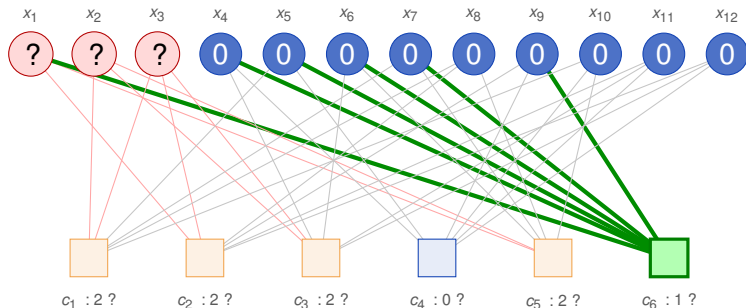
Itération 1 : c_4 n'a qu'un seul voisin effacé : la parité force sa valeur.

Exemple : 4 effacements, décodage réussi



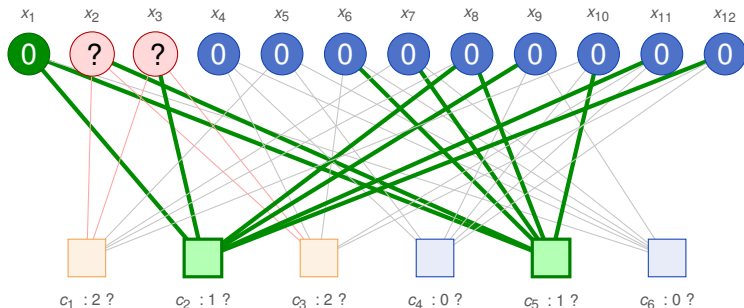
x_4 résolu (= 0) et propagé aux contrôles voisins. Effacements restants : 3.

Exemple : 4 effacements, décodage réussi



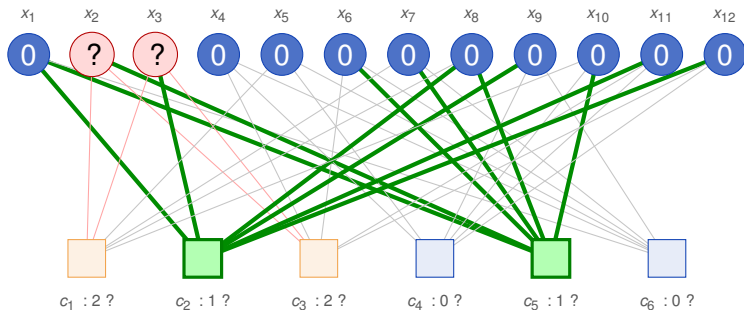
Itération 2 : c_6 n'a qu'un seul voisin effacé : la parité force sa valeur.

Exemple : 4 effacements, décodage réussi



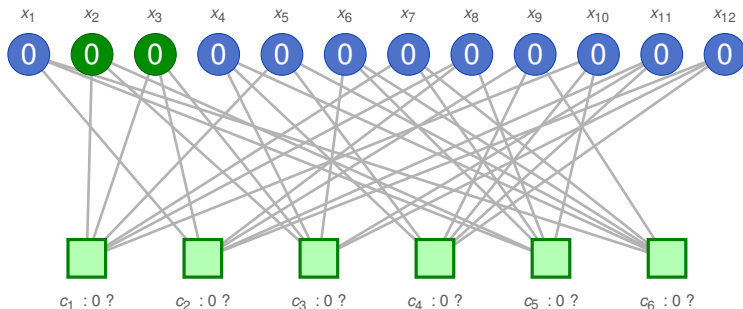
x_1 résolu (= 0) et propagé aux contrôles voisins. Effacements restants : 2.

Exemple : 4 effacements, décodage réussi



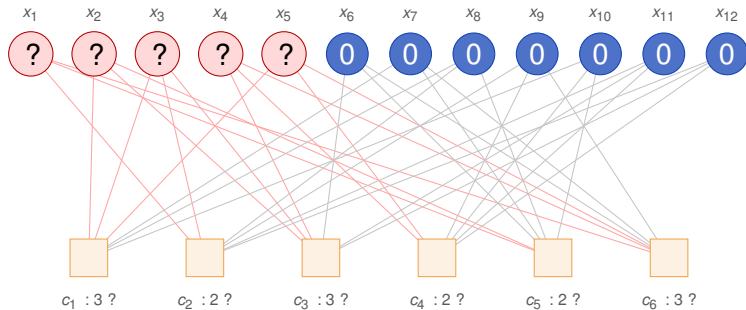
Itération 3 : c_2, c_5 ont chacun un seul voisin effacé : la parité force leur valeur.

Exemple : 4 effacements, décodage réussi



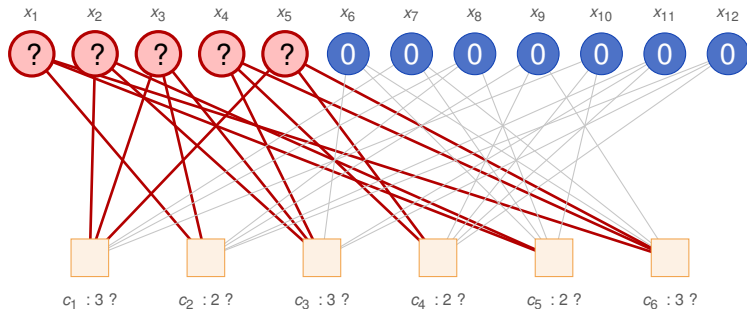
Décodage terminé : tous les effacements sont résolus.

Exemple : 5 effacements, décodage bloqué



Motif de 5 effacements $x_1, \dots, x_5$: aucun contrôle n'a exactement un voisin effacé.

Exemple : 5 effacements, décodage bloqué



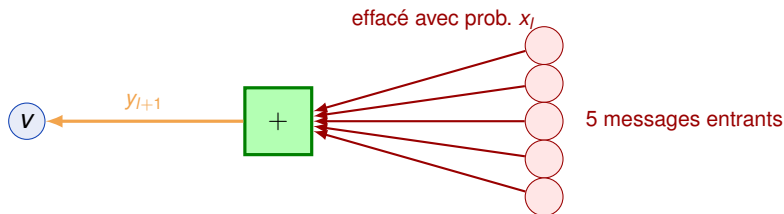
Décodage bloqué : l'ensemble $\{x_1, \dots, x_5\}$ est un "stopping set" : chaque noeud de parité voisin a au moins 2 effacements.

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité**
- 6 Seuils et simulations

Messages et évolution de densité

Pour n grand, les messages entrants d'un nœud sont indépendants. On suit x_l = probabilité qu'un message variable $\rightarrow$ contrôle soit un effacement.

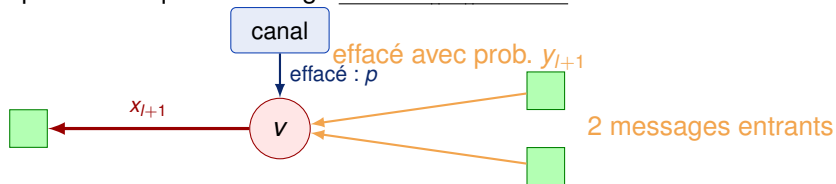


Contrôle $\rightarrow$ **variable** : sortie effacée dès qu'**une** des 5 entrées l'est

$$y_{l+1} = 1 - (1 - x_l)^5 \quad (= 1 - (1 - x_l)^{d_c - 1})$$

Messages et évolution de densité

Pour n grand, les messages entrants d'un nœud sont indépendants. On suit x_l = probabilité qu'un message variable $\rightarrow$ contrôle soit un effacement.



Variable $\rightarrow$ **contrôle** : effacée seulement si le canal **et** les 2 autres entrées le sont

$$x_{l+1} = p y_{l+1}^2 \quad (= p y_{l+1}^{d_v-1})$$

Messages et évolution de densité

Composition : une itération

$$x_{l+1} = p \left(1 - (1 - x_l)^5 \right)^2 =: f_p(x_l), \quad x_0 = p$$

Cas général (ensembles irréguliers)

$$x_{l+1} = p \lambda (1 - \rho(1 - x_l))$$

Une seule récurrence scalaire suffit : l'évolution de densité est **exacte** sur le BEC.

Seuil de décodage

Définition

$p^* = \sup\{p : x_I \rightarrow 0\}$; c'est la plus grande probabilité d'effacement corrigée quand $n \rightarrow \infty$.

- $x_I \rightarrow 0$ si et seulement si $f_p(x) < x$ pour tout $x \in (0, p]$ (aucun point fixe non nul).
- Pour (3, 6) :

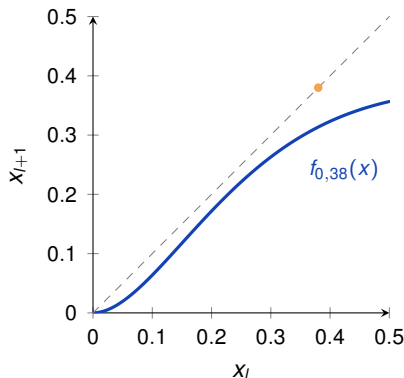
$$p^* = \min_{x \in (0,1]} \frac{x}{(1 - (1 - x)^5)^2} \approx \mathbf{0,4294}$$

- Capacité du BEC à $R = \frac{1}{2}$: $p_{\text{Shannon}} = 0.5$.

Lecture

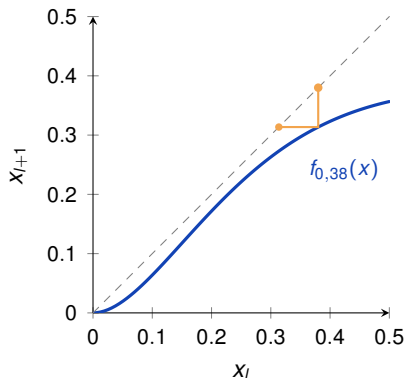
Sous p^* : l'effacement disparaît. Au-dessus : un **point fixe** $x^* > 0$ bloque le décodage.

En dessous du seuil : $p = 0.38 < p^*$



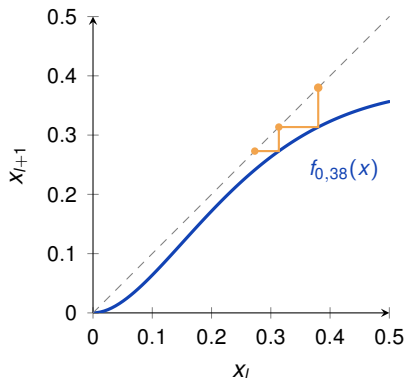
La courbe f_p reste sous la diagonale : pas de point fixe non nul.

En dessous du seuil : $p = 0.38 < p^*$



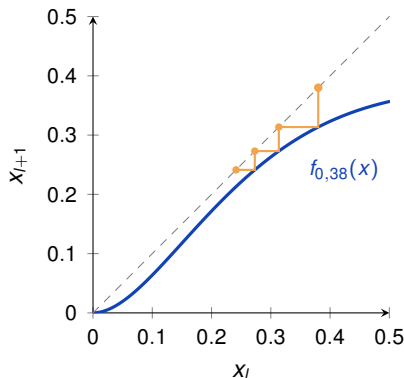
Itération $l = 1$: on monte sur f_p , puis on redescend sur la diagonale.

En dessous du seuil : $p = 0.38 < p^*$



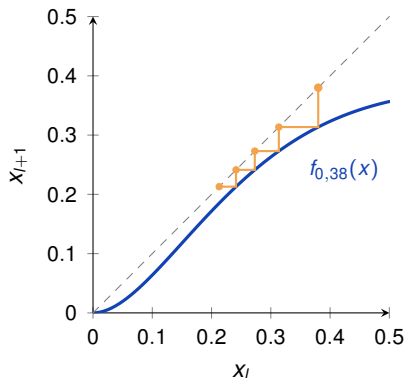
Itération $l = 2$: on monte sur f_p , puis on redescend sur la diagonale.

En dessous du seuil : $p = 0.38 < p^*$



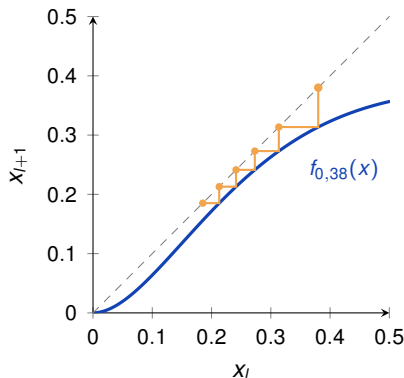
Itération $I = 3$: on monte sur f_p , puis on redescend sur la diagonale.

En dessous du seuil : $p = 0.38 < p^*$



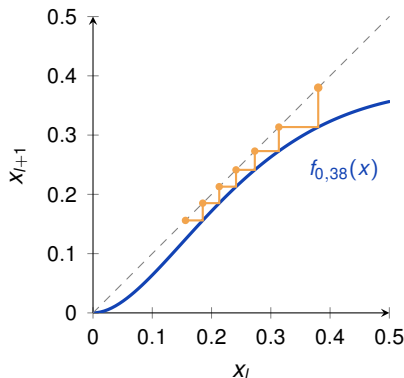
Itération $l = 4$: on monte sur f_p , puis on redescend sur la diagonale.

En dessous du seuil : $p = 0.38 < p^*$



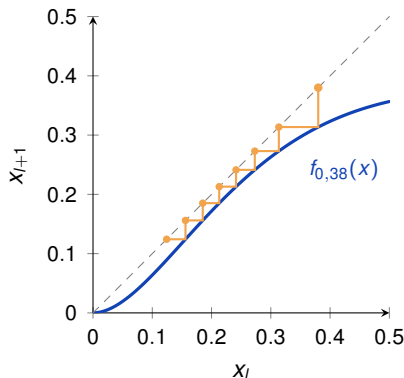
Itération $l = 5$: on monte sur f_p , puis on redescend sur la diagonale.

En dessous du seuil : $p = 0.38 < p^*$



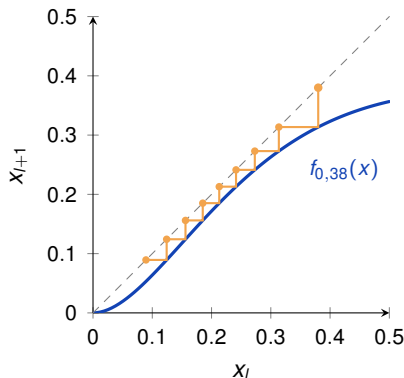
Itération $l = 6$: on monte sur f_p , puis on redescend sur la diagonale.

En dessous du seuil : $p = 0.38 < p^*$



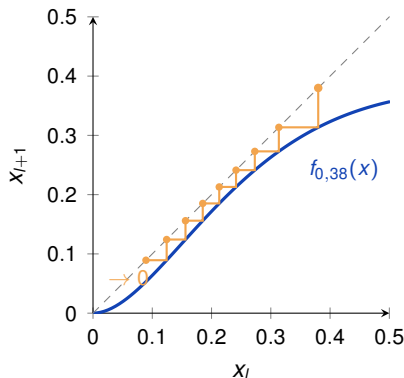
Itération $l = 7$: on monte sur f_p , puis on redescend sur la diagonale.

En dessous du seuil : $p = 0.38 < p^*$



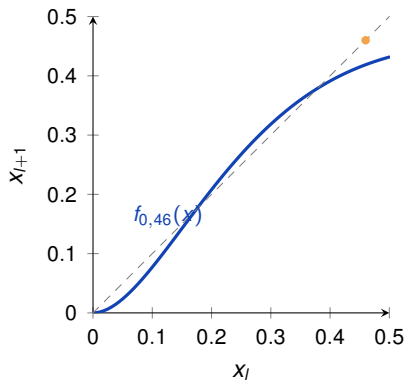
Itération $l = 8$: on monte sur f_p , puis on redescend sur la diagonale.

En dessous du seuil : $p = 0.38 < p^*$



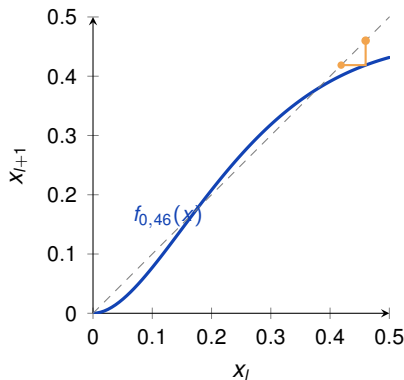
La suite x_i décroît jusqu'à 0 : tous les effacements sont corrigés.

Au-dessus du seuil : $p = 0,46 > p^*$



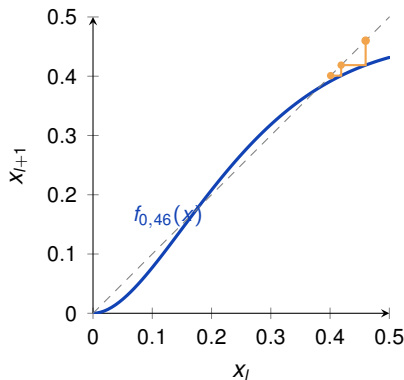
Ici f_p coupe la diagonale.

Au-dessus du seuil : $p = 0,46 > p^*$



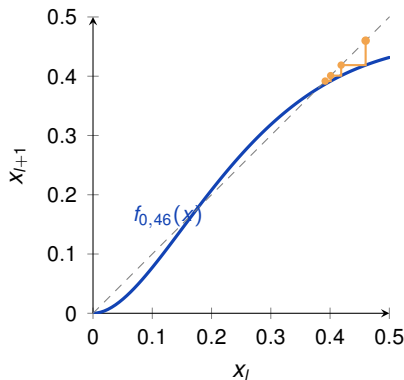
Itération $l = 1$: la suite x_l se rapproche d'un point fixe.

Au-dessus du seuil : $p = 0,46 > p^*$



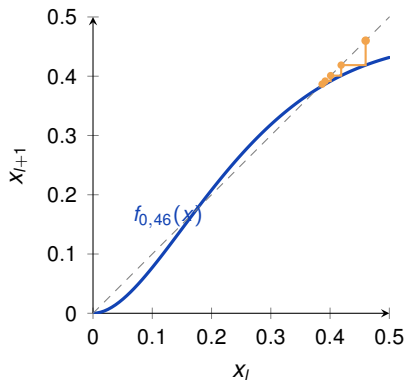
Itération $l = 2$: la suite x_l se rapproche d'un point fixe.

Au-dessus du seuil : $p = 0,46 > p^*$



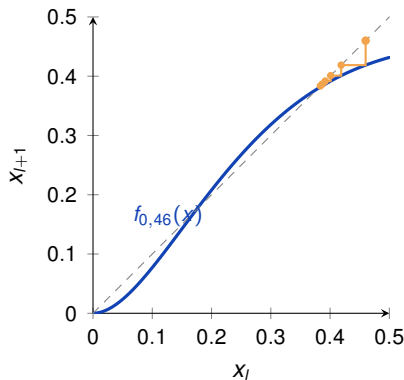
Itération $l = 3$: la suite x_l se rapproche d'un point fixe.

Au-dessus du seuil : $p = 0,46 > p^*$



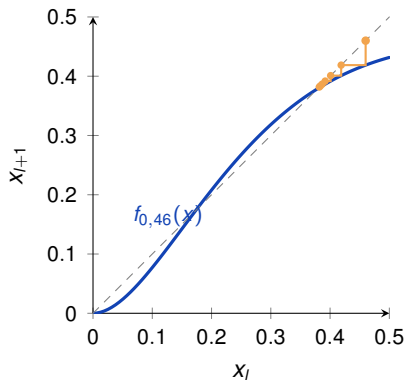
Itération $l = 4$: la suite x_l se rapproche d'un point fixe.

Au-dessus du seuil : $p = 0,46 > p^*$



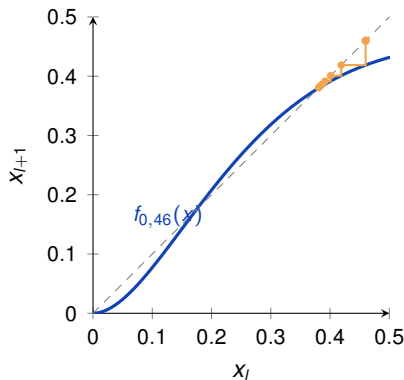
Itération $l = 5$: la suite x_l se rapproche d'un point fixe.

Au-dessus du seuil : $p = 0,46 > p^*$



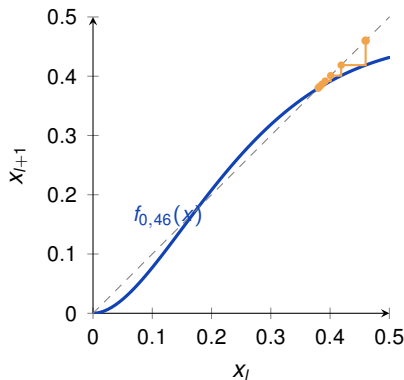
Itération $l = 6$: la suite x_l se rapproche d'un point fixe.

Au-dessus du seuil : $p = 0,46 > p^*$



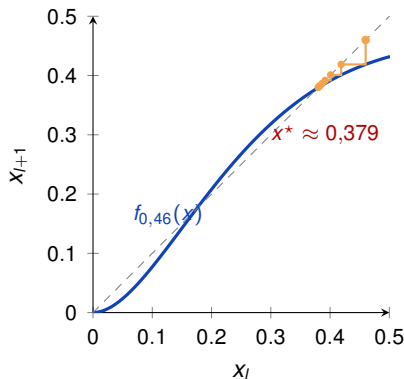
Itération $l = 7$: la suite x_l se rapproche d'un point fixe.

Au-dessus du seuil : $p = 0,46 > p^*$



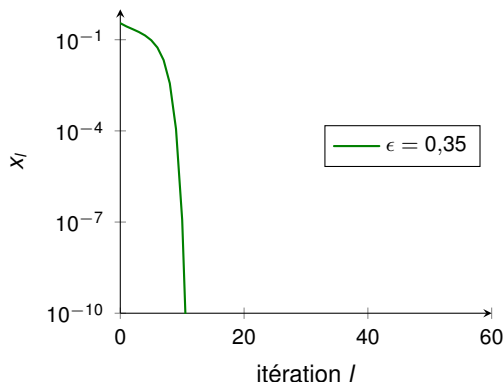
Itération $l = 8$: la suite x_l se rapproche d'un point fixe.

Au-dessus du seuil : $p = 0,46 > p^*$



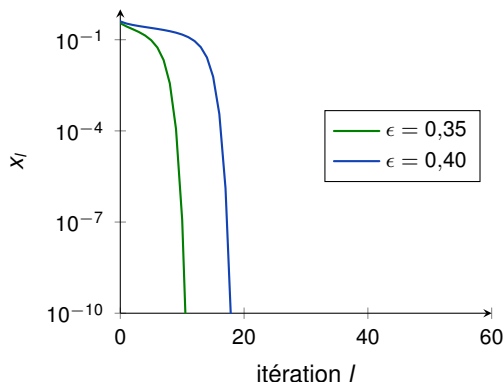
La suite converge vers $x^* > 0$: une fraction des messages reste effacée **pour toujours**.

Évolution de x_l selon p (code (3, 6))



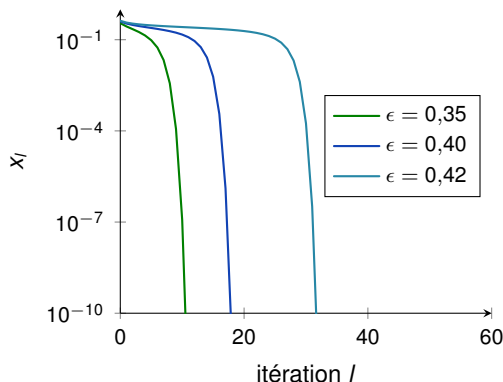
Sous $p < p^*$: décroissance (super-)exponentielle vers 0 ; plus on s'approche du seuil, plus la convergence ralentit. Au-dessus : plateau en $x^* > 0$.

Évolution de x_l selon p (code (3, 6))



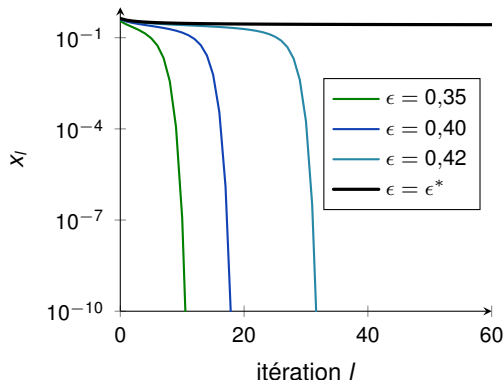
Sous $p < p^*$: décroissance (super-)exponentielle vers 0 ; plus on s'approche du seuil, plus la convergence ralentit. Au-dessus : plateau en $x^* > 0$.

Évolution de x_l selon p (code (3, 6))



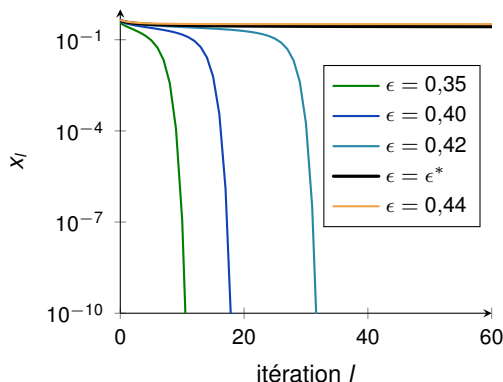
Sous $p < p^*$: décroissance (super-)exponentielle vers 0 ; plus on s'approche du seuil, plus la convergence ralentit. Au-dessus : plateau en $x^* > 0$.

Évolution de x_l selon p (code (3, 6))



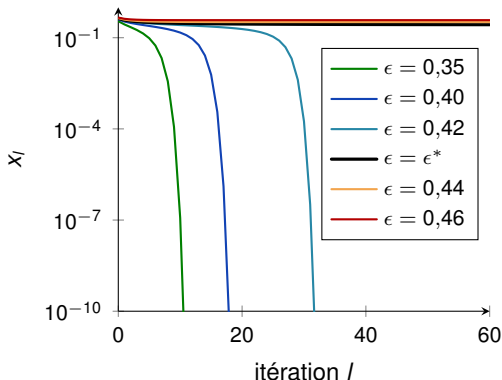
Sous $p < p^*$: décroissance (super-)exponentielle vers 0 ; plus on s'approche du seuil, plus la convergence ralentit. Au-dessus : plateau en $x^* > 0$.

Évolution de x_l selon p (code (3, 6))



Sous $p < p^*$: décroissance (super-)exponentielle vers 0 ; plus on s'approche du seuil, plus la convergence ralentit. Au-dessus : plateau en $x^* > 0$.

Évolution de x_l selon p (code (3, 6))



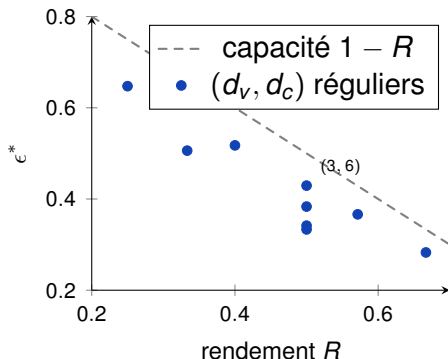
Sous $p < p^*$: décroissance (super-)exponentielle vers 0 ; plus on s'approche du seuil, plus la convergence ralentit. Au-dessus : plateau en $x^* > 0$.

Plan

- 1 Introduction générale
- 2 Rappels sur de codage / définitions
- 3 Codes Linéaires (binaires) en blocs
- 4 LDPC
- 5 Évolution de densité
- 6 Seuils et simulations**

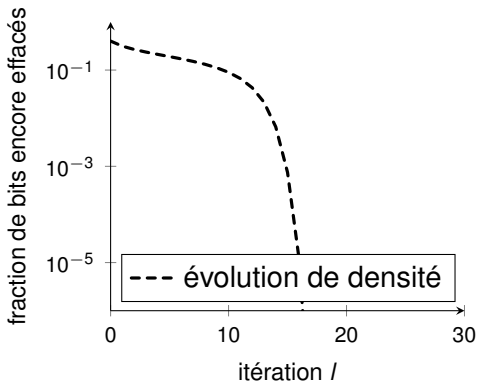
Seuils des ensembles réguliers vs capacité

(d_v, d_c)	R	p^*
(3, 4)	0,25	0,6474
(3, 5)	0,40	0,5176
(3, 6)	0,50	0,4294
(4, 8)	0,50	0,3834
(5, 10)	0,50	0,3416
(3, 7)	0,57	0,3664
(3, 9)	0,67	0,2828



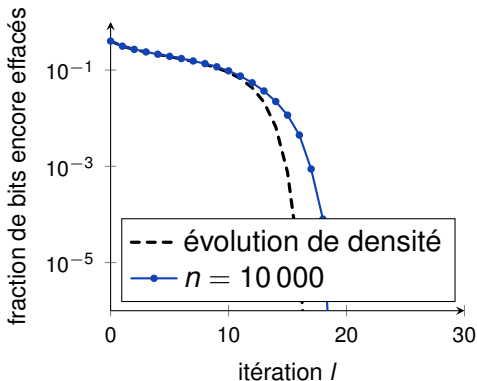
Les codes réguliers restent à distance de la capacité ; au équilibre $R = \frac{1}{2}$, (3, 6) est le meilleur des trois. Les ensembles **irréguliers** optimisés s'en rapprochent arbitrairement.

Simulation sur graphes aléatoires



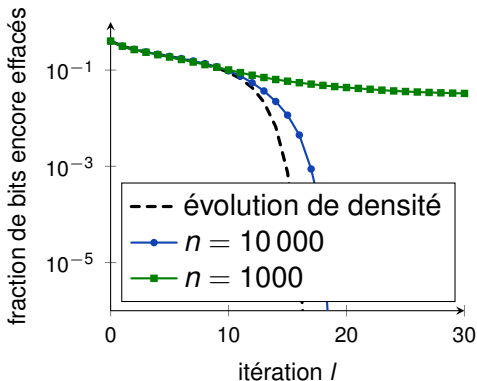
$p = 0,40$; passage de messages sur des graphes $(3, 6)$ aléatoires (moyennes sur 10, 100 et 300 graphes). $n = 10\,000$ suit la théorie puis s'en écarte (taille finie) ; pour $n = 1000$ et 200 : **plateaux** dus aux ensembles d'arrêt.

Simulation sur graphes aléatoires



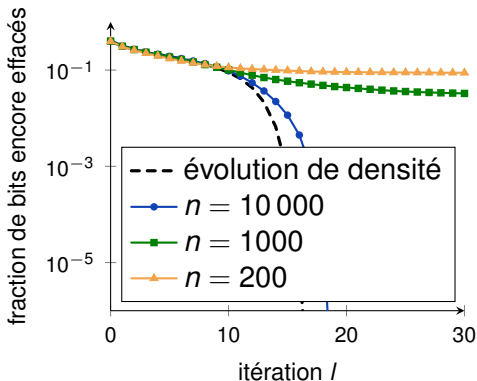
$p = 0,40$; passage de messages sur des graphes $(3, 6)$ aléatoires (moyennes sur 10, 100 et 300 graphes). $n = 10\,000$ suit la théorie puis s'en écarte (taille finie) ; pour $n = 1000$ et 200 : **plateaux** dus aux ensembles d'arrêt.

Simulation sur graphes aléatoires



$p = 0,40$; passage de messages sur des graphes $(3, 6)$ aléatoires (moyennes sur 10, 100 et 300 graphes). $n = 10\,000$ suit la théorie puis s'en écarte (taille finie) ; pour $n = 1\,000$ et 200 : **plateaux** dus aux ensembles d'arrêt.

Simulation sur graphes aléatoires



$p = 0,40$; passage de messages sur des graphes $(3, 6)$ aléatoires (moyennes sur 10, 100 et 300 graphes). $n = 10\,000$ suit la théorie puis s'en écarte (taille finie) ; pour $n = 1\,000$ et 200 : **plateaux** dus aux ensembles d'arrêt.

Ce qu'il faut retenir

Évolution de densité LDPC / BEC

- Le décodage par épuration = passage de messages : un contrôle n'a plus qu'un voisin effacé $\Rightarrow$ il le résout.
- Pour $n \rightarrow \infty$: $x_{l+1} = p \lambda(1 - \rho(1 - x_l))$, suite scalaire **exacte**.
- **Seuil** p^* : (3, 6) donne 0,4294 contre la capacité 0,5.

Limites

À n fini, les **ensembles d'arrêt** bloquent le décodage même sous le seuil (cf. $n = 200$, $n = 1000$).