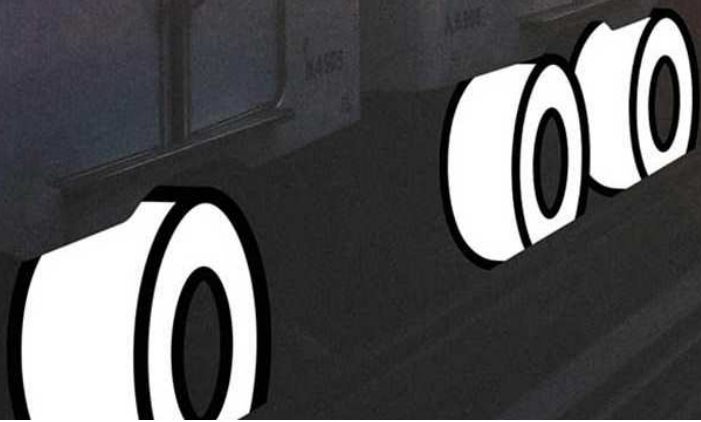
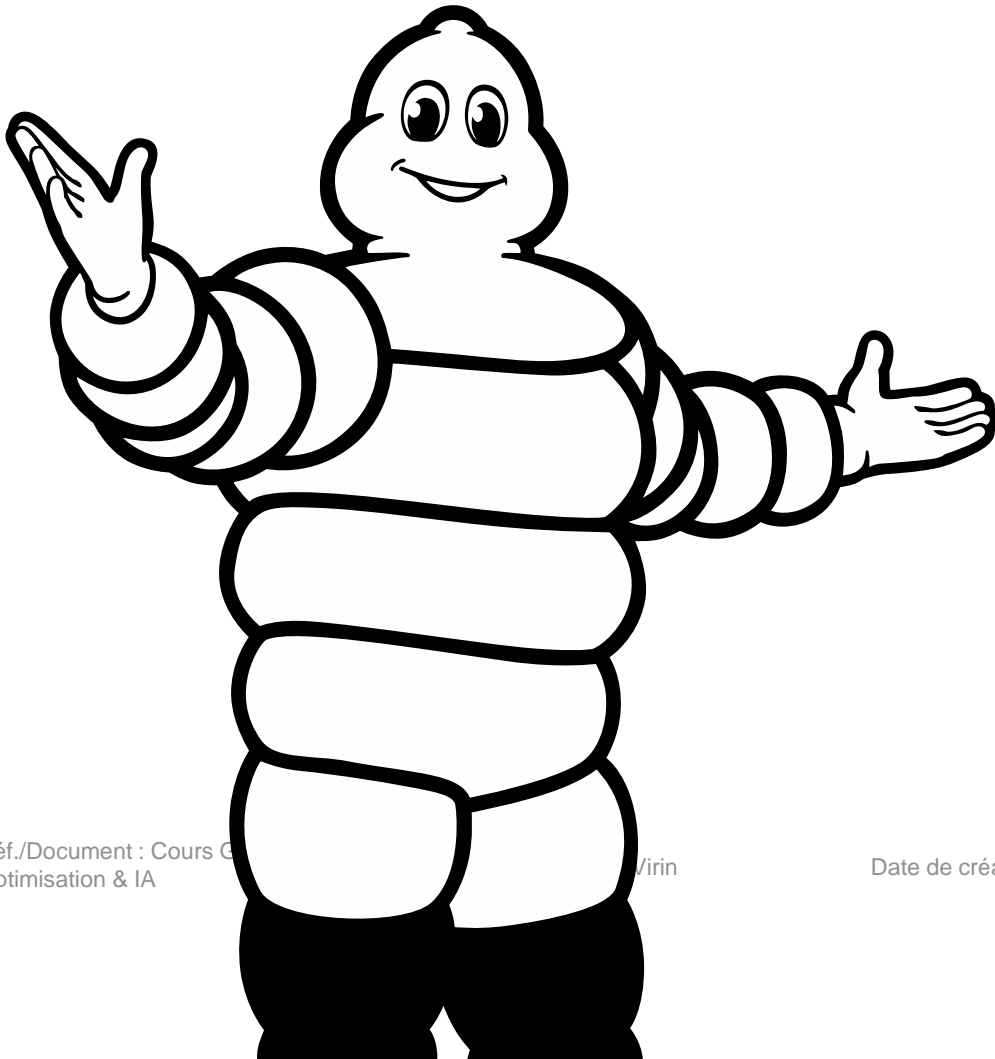


L'IA AU SERVICE DE L'OPTIMISATION

22/11/2021



Rappels et Motivations



- 1. Qu'est-ce-que l'Optimisation?***
- 2. L'IA au Service de l'Optimisation***



Qu'est ce que l'Optimisation?

⊙ Optimiser:

Trouver le (les) minimum (-a) ou le (les) maximum (-a) d'une fonction en la présence ou non de contraintes

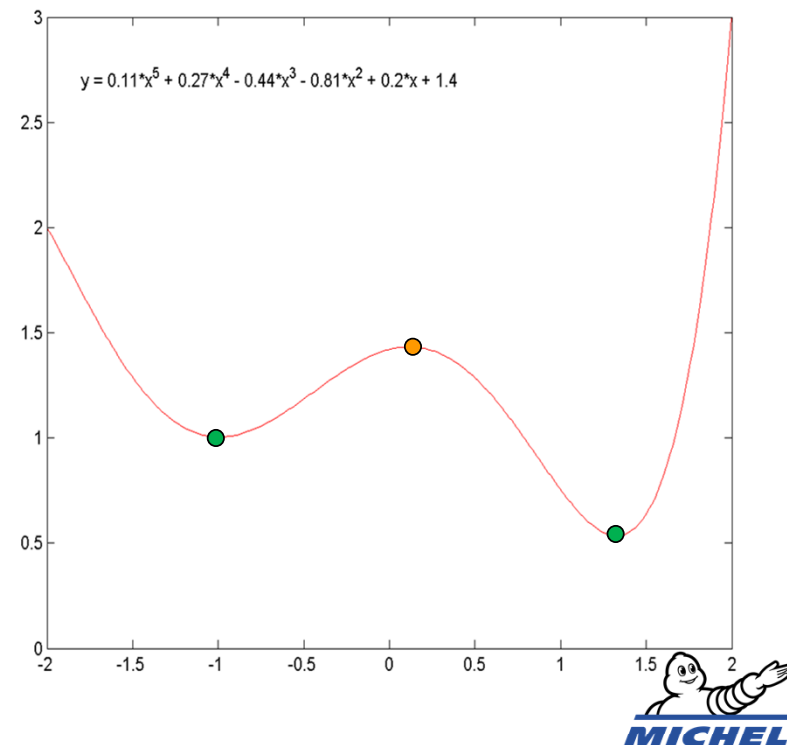
$$\forall x \in C, f(x) \geq f(x^*)$$

$$\forall x \in C, f(x) \leq f(x^*)$$

C ensemble de solutions admissibles

"True Optimization is the revolutionary contribution of modern research to decision processes"

Georges Dantzig (1914-2005)





Différentes Classes de Problèmes

⊙ Problèmes continus:

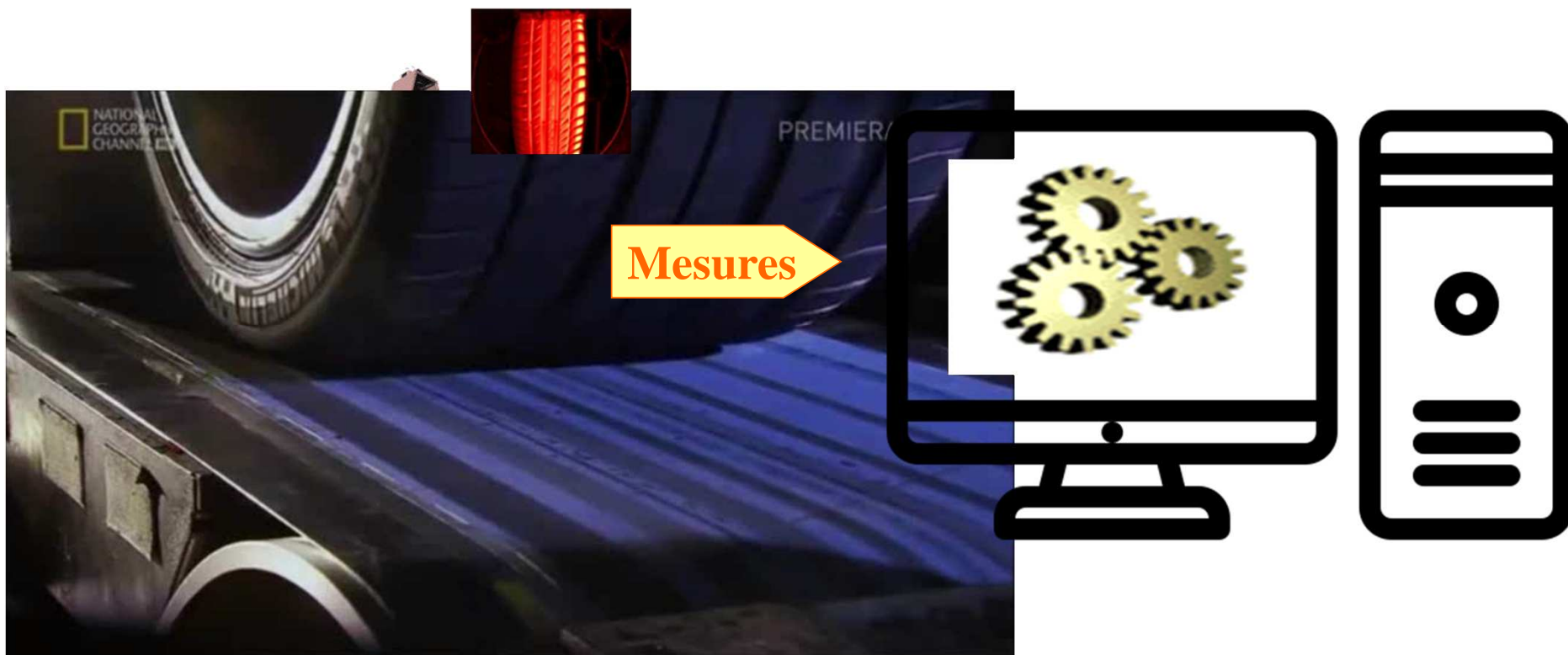
$$\min_{(x,y) \in C} f(x,y) = (x \exp(-2y) + x^2)$$
$$C = [-10^8, 10^8] \times [2, 3]$$

⊙ Problèmes non différentiables:

$$\min_{(x,y) \in \mathbb{R}^2} f(x,y) = \max \left\{ -x - y, -x - y + (x^2 + y^2 - 1) \right\}$$



Différentes Classes de Problèmes

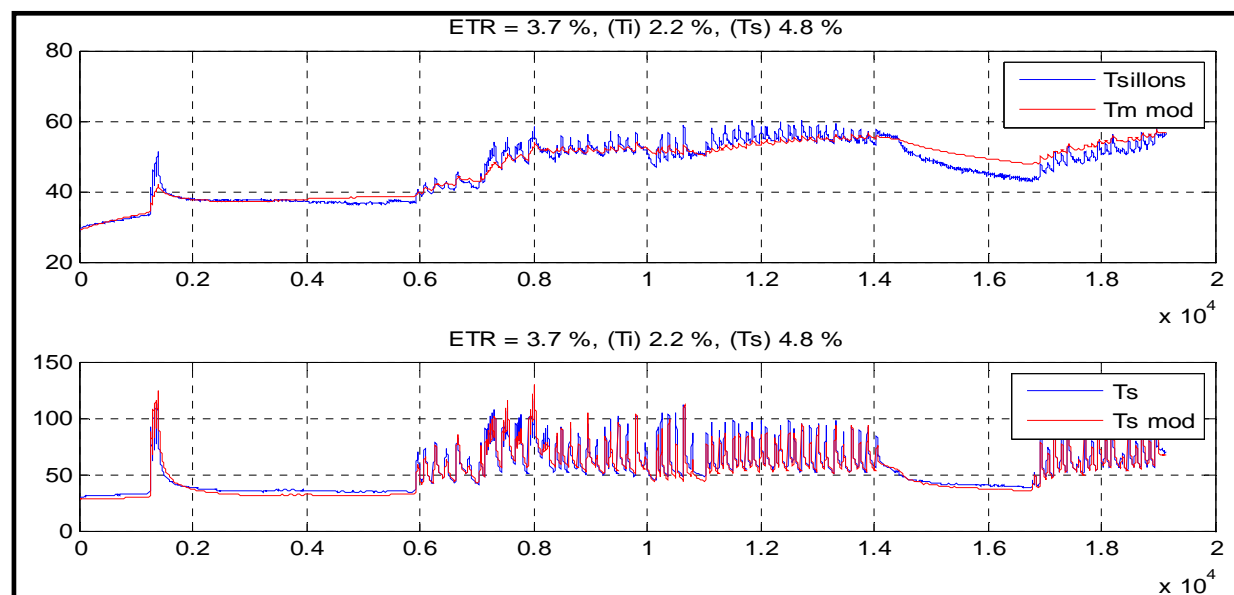
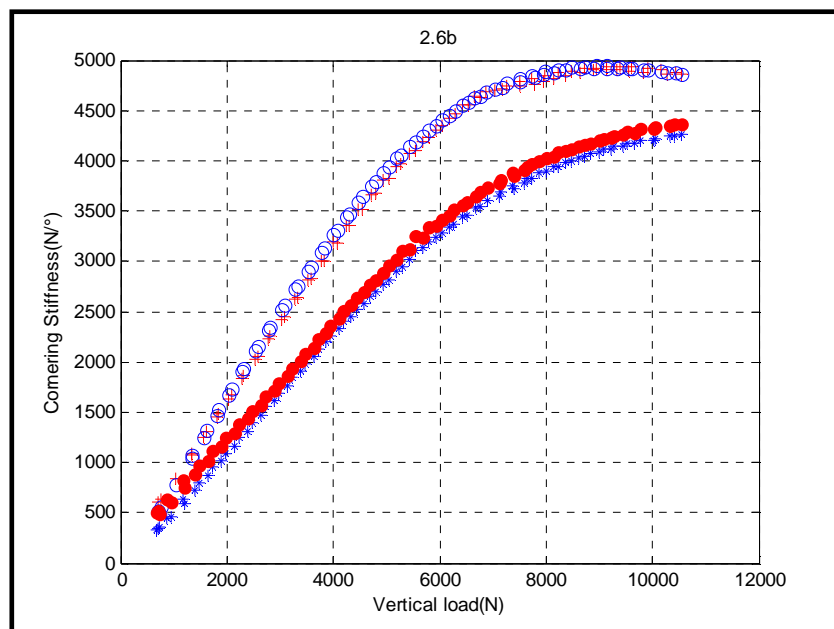


5





Différentes Classes de Problèmes





Différentes Classes de Problèmes

⊙ Problèmes continus:

$$\min_{(x,y) \in C} f(x,y) = (x \exp(-2y) + x^2)$$
$$C = [-10^8, 10^8] \times [2, 3]$$

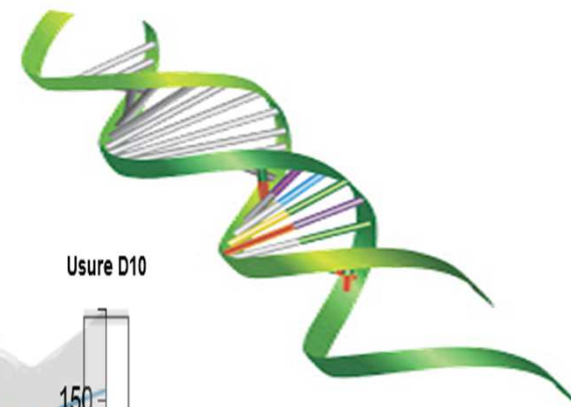
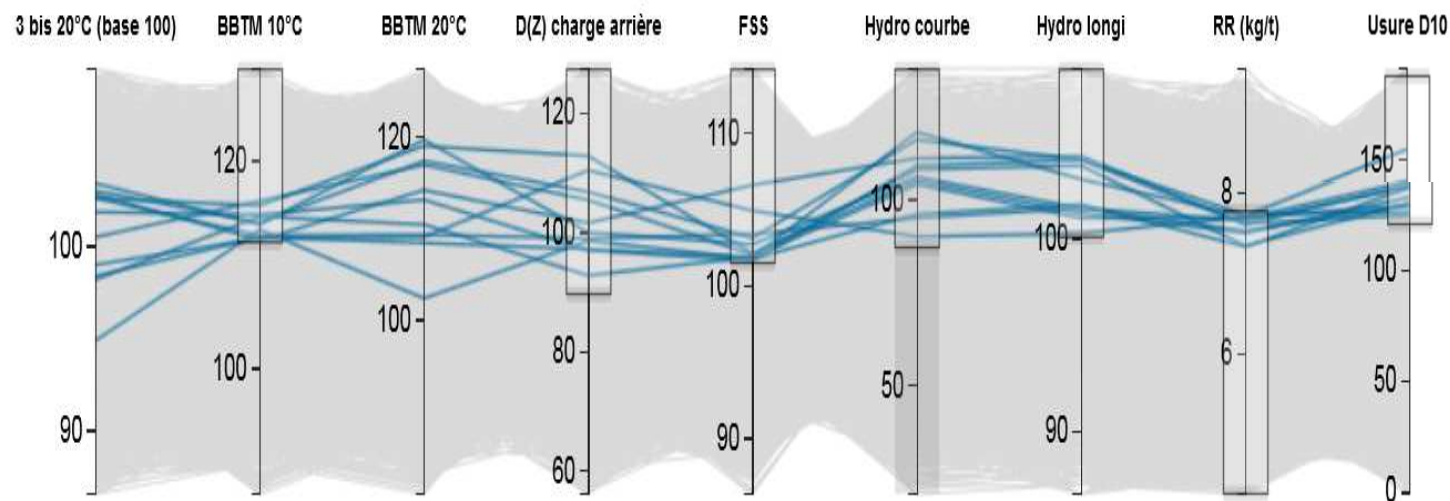
⊙ Problèmes non différentiables:

$$\min_{(x,y) \in \mathbb{R}^2} f(x,y) = \max \left\{ -x - y, -x - y + (x^2 + y^2 - 1) \right\}$$

⊙ Problèmes multi-objectifs



Différentes Classes de Problèmes





Différentes Classes de Problèmes

⊙ Problèmes continus:

$$\min_{(x,y) \in C} f(x,y) = (x \exp(-2y) + x^2)$$

$$C = [-10^8, 10^8] \times [2, 3]$$

⊙ Problèmes non différentiables:

$$\min_{(x,y) \in \mathbb{R}^2} f(x,y) = \max \left\{ -x - y, -x - y + (x^2 + y^2 - 1) \right\}$$

⊙ Problèmes multi-objectifs

⊙ Optimisation Combinatoire (plus court chemin, ordonnancement, planification,...)

$$\begin{aligned} \min \quad & c^T x \\ & Ax \leq b \\ & x \in \{0,1\} \end{aligned}$$



Différentes Classes de Problèmes

⊙ Problèmes continus:

$$\min_{(x,y) \in C} f(x,y) = (x \exp(-2y) + x^2)$$

$$C = [-10^8, 10^8] \times [2, 3]$$

⊙ Problèmes non différentiables:

$$\min_{(x,y) \in \mathbb{R}^2} f(x,y) = \max \left\{ -x - y, -x - y + (x^2 + y^2 - 1) \right\}$$

⊙ Problèmes multi-objectifs

⊙ Optimisation Combinatoire (plus court chemin, ordonnancement, planification,...)

⊙ Optimisation Stochastique

$$(\mathcal{P}) \begin{cases} \text{"min"} f(x, \xi) \\ c(x, \xi) \leq 0 \\ x \in X \subset \mathbb{R}^n, \xi \text{ une variable alatoire de } (\Omega, \Sigma, P) \end{cases}$$

$$\min c^T x$$





L'IA au Service de l'Optimisation

- ⊙ **L'Optimisation s'appuie sur la connaissance de modélisation de fonctions objectifs et de contraintes**

$$\forall x \in C, f(x) \geq f(x^*)$$

$$\forall x \in C, f(x) \leq f(x^*)$$

C ensemble de solutions admissibles

- ⊙ **Comment faire quand:**

- La fonction coût est boîte noire
- Pas de connaissance directe du gradient
- L'évaluation de la fonction objectif est coûteuse



Optimisation Bayesienne



L'IA au Service de l'Optimisation

⊙ L'Optimisation s'appuie sur la connaissance de modélisation de fonctions objectifs et de contraintes

$$\forall x \in C, f(x) \geq f(x^*)$$

$$\forall x \in C, f(x) \leq f(x^*)$$

C ensemble de solutions admissibles

⊙ Comment faire quand:

- Le but est de définir une séquence de décisions optimales impliquant:
 - Un grand nombre de paramètres
 - Beaucoup de critères à optimiser en même temps



Apprentissage par Renforcement

Optimisation Bayésienne

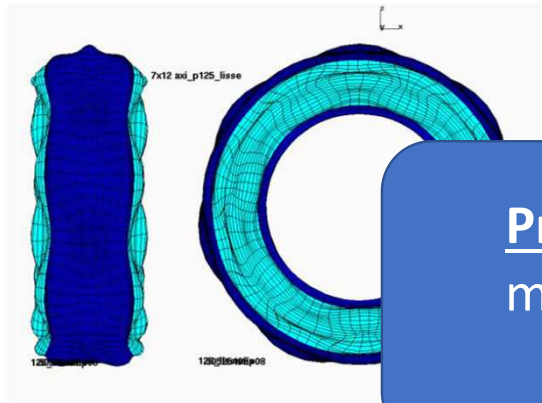


- 1. Quelques Exemples***
- 2. Processus Gaussien***
- 3. Planification d'Expériences Numériques***
- 4. Optimisation Bayésienne***



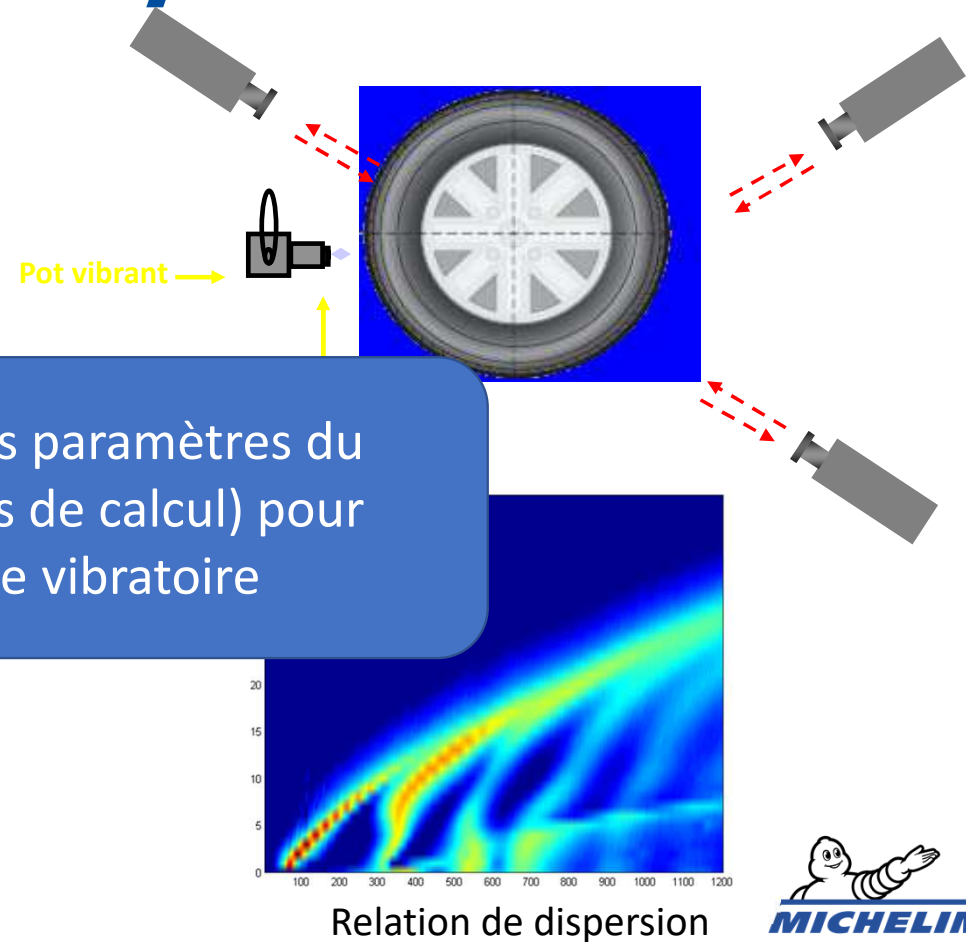
Quelques Exemples

Calibration de Modèle:



Réponse Vibratoire

Problématique: Ajuster les paramètres du modèle (coûteux en temps de calcul) pour représenter la réponse vibratoire



Diapositive 14

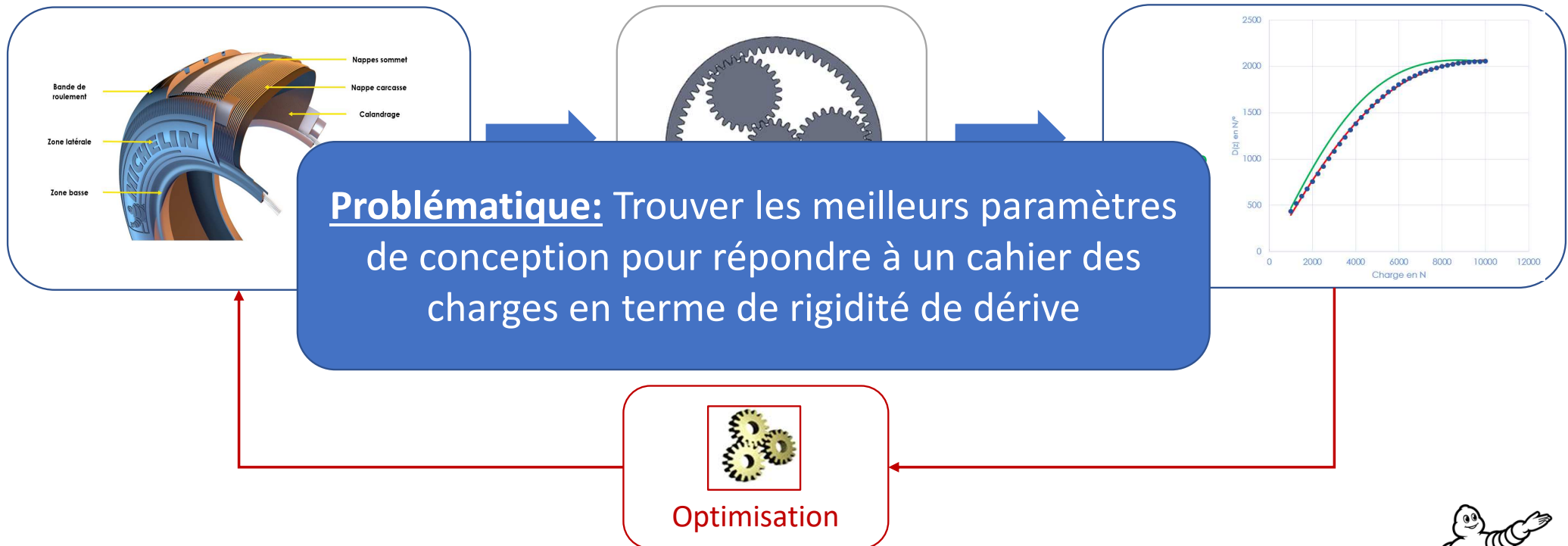
TV1

Teddy Virin; 21/11/2021



Quelques Exemples

Optimisation de Conception:





Processus Gaussien

Quelques Rappels

⊙ Densité de la loi normale en dimension 1:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

⊙ Densité de la loi normale en dimension N :

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{N/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x - \mu)^\top \Sigma^{-1} (x - \mu)\right]$$



Processus Gaussien

Quelques Rappels

- ⊙ **Variable aléatoire vectorielle:**

X est une v.a., on associe à chaque événement ω n éléments de \mathbb{R}

$$\omega \rightarrow [X_1(\omega), X_2(\omega), \dots, X_n(\omega)]$$

- ⊙ **Fonction aléatoire :**

v.a. vectorielle avec une infinité de composantes

$$X(t), X(y)$$

- ⊙ **Champ réel aléatoire :**

fonction aléatoire dans un espace à plusieurs dimensions

$$X(y, z), X(t, y, z)$$



Processus Stochastique

⊙ Processus stochastique :

- Un processus stochastique $X = (X_t)_{t \in T}$ est une famille de variables aléatoires X_t indexée par un ensemble T
- Un processus dépend de deux paramètres : $X_t(\omega)$ dépend de t et de l'aléatoire ω
Si t est fixé, $X_t(\omega)$ est une variable aléatoire
Si ω est fixé, $X_t(\omega)$ est une fonction à valeurs réelles
- Etant donné un processus stochastique $(X_t)_{t \in T}$, les lois finies dimensionnelles de X sont les lois de tous les vecteurs $(X_{t_1}, X_{t_2}, \dots, X_{t_n})$ pour $t_1, t_2, \dots, t_n \in T$ et $n \in \mathbb{N}$



Processus Gaussien

◉ Processus Gaussien:

- **Un processus est dit gaussien si toutes ses lois finies dimensionnelles sont gaussiennes**
 $(\forall n \in \mathbb{N}, \forall t_1, \dots, t_n \in T)$
- **Autrement dit, $X = (X_t)_{t \in T}$ est gaussien si toute combinaison linéaire $a_1 X_{t_1} + \dots + a_n X_{t_n}$ suit une loi gaussienne** $(\forall n \in \mathbb{N}, \forall t_1, \dots, t_n \in T, a_1, \dots, a_n \in \mathbb{R})$

- **Il est entièrement défini par:**

- une fonction moyenne:

$$m(t) = \mathbb{E}[X_t]$$

- une fonction de covariance:

$$k(t, t') = \text{Cov}(X_t, X_{t'}) = \mathbb{E}[(X_t - m(t)) (X_{t'} - m(t'))^T]$$

- **Il s'écrit:**

$$X(t) \sim PG(m(t), k(t, t'))$$



Processus Gaussien

◉ Exemple:

- On considère un processus gaussien 1D $X(t) \sim PG(m(t), k(t, t'))$ avec:

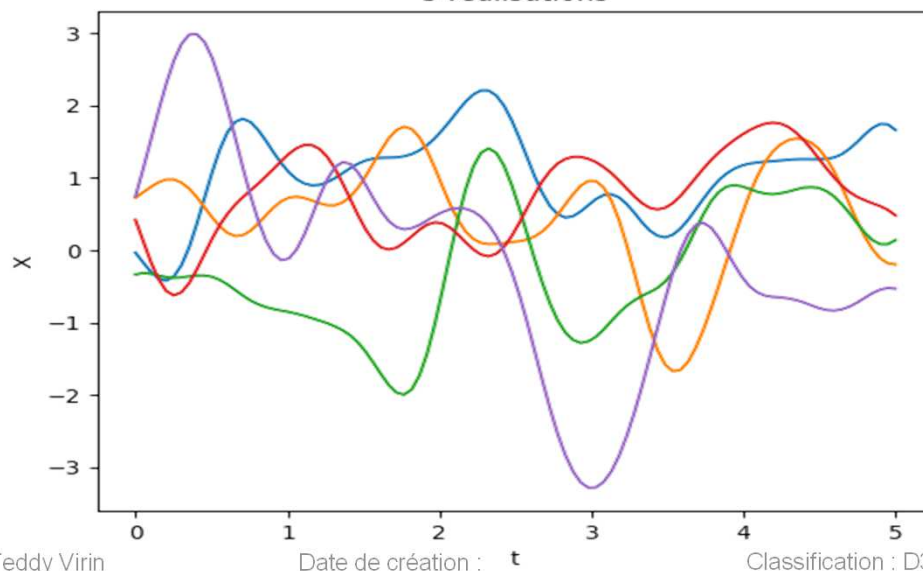
- une fonction moyenne nulle:

$$m(t) = 0$$

- une fonction de covariance correspondant au noyau (kernel) exponentiel au carré:

$$k(t, t') = \sigma^2 \exp\left(-\frac{(t-t')^2}{2l^2}\right) \text{ avec } l = 1 \text{ et } \sigma^2 = 1$$

5 réalisations

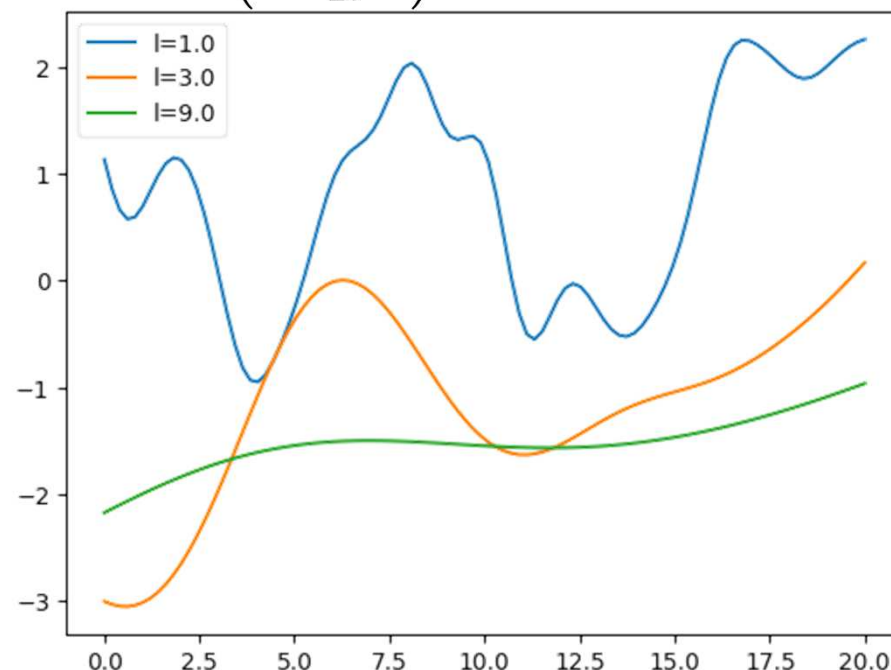
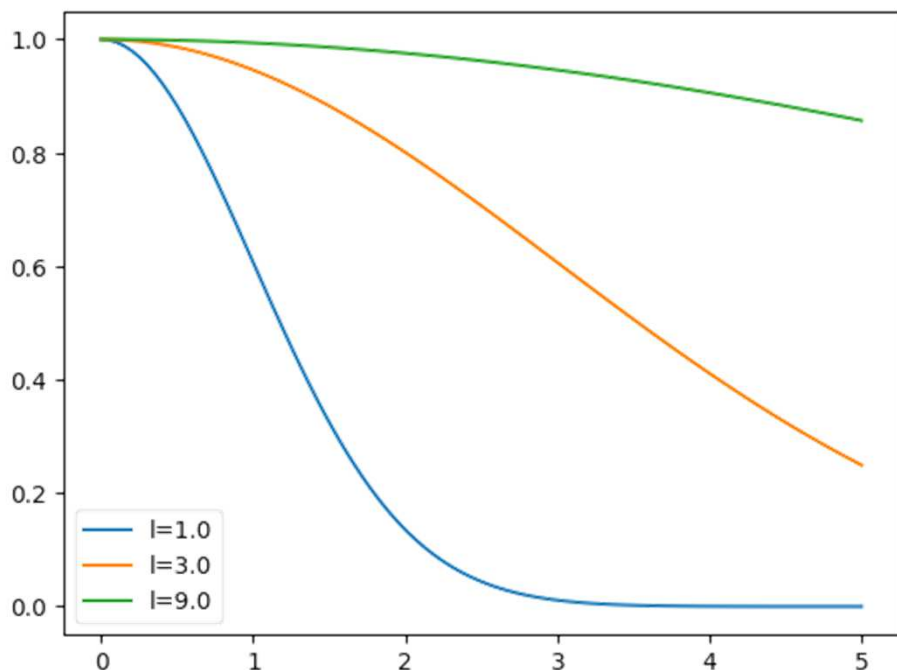




Fonction de Covariance

◉ Il existe plusieurs fonctions de covariance couramment utilisées:

- **Covariance exponentielle au carré:** $k(t, t') = \sigma^2 \exp\left(-\frac{(t-t')^2}{2l^2}\right)$



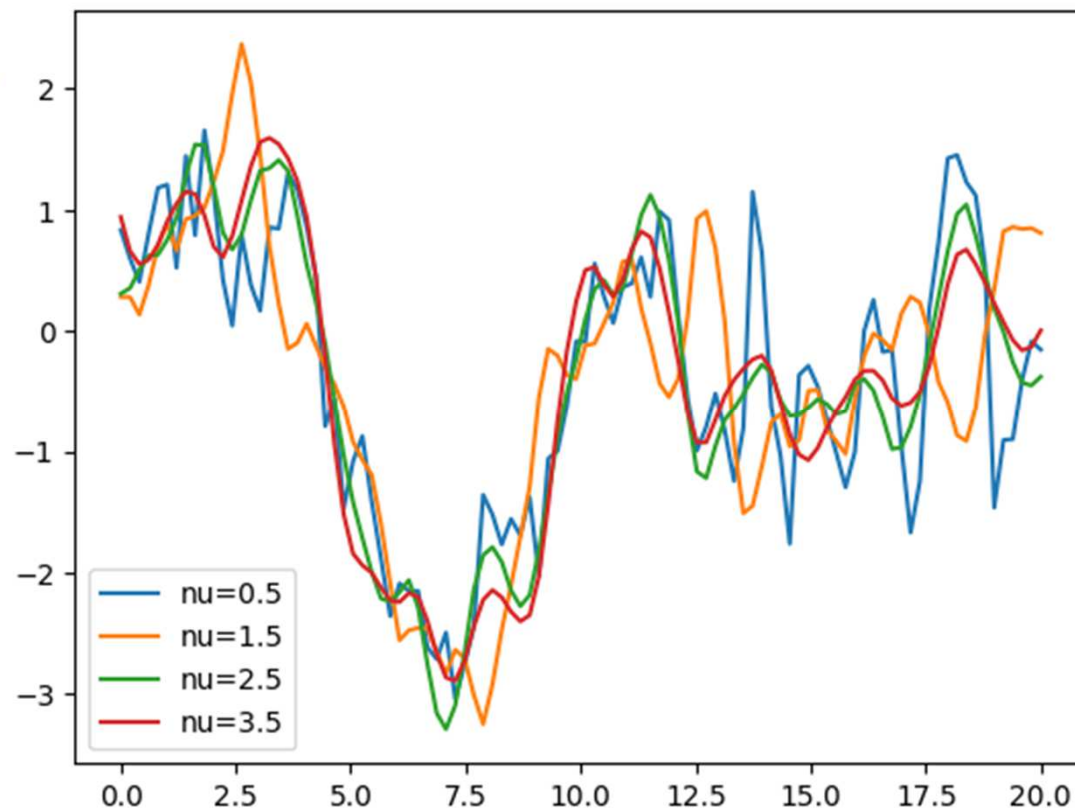
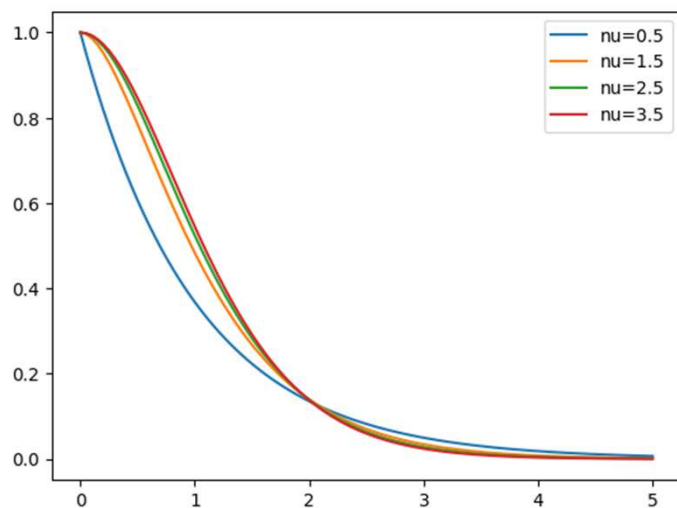
Fonction indéfiniment différentiable → Réalisations sont lisses



Fonction de Covariance

◉ Il existe plusieurs fonctions de covariance couramment utilisées:

- **Classe Matern:** $k(\mathbf{t}, \mathbf{t}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}(\mathbf{t}-\mathbf{t}')}{l} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}(\mathbf{t}-\mathbf{t}')}{l} \right)$
- Le processus gaussien est k fois différentiable
- On obtient la fonction de covariance en
- Le plus souvent, on utilise la classe M

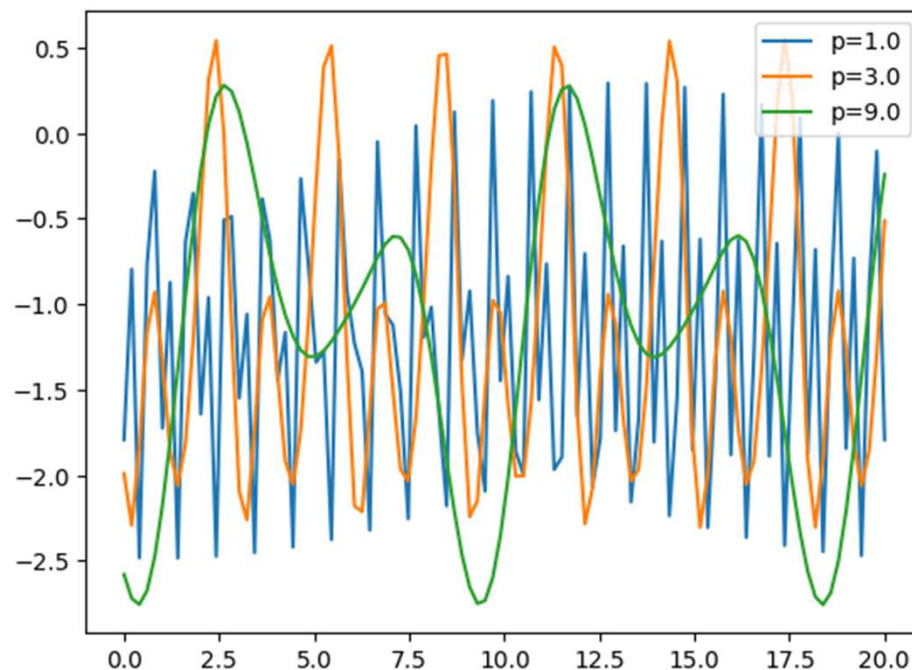
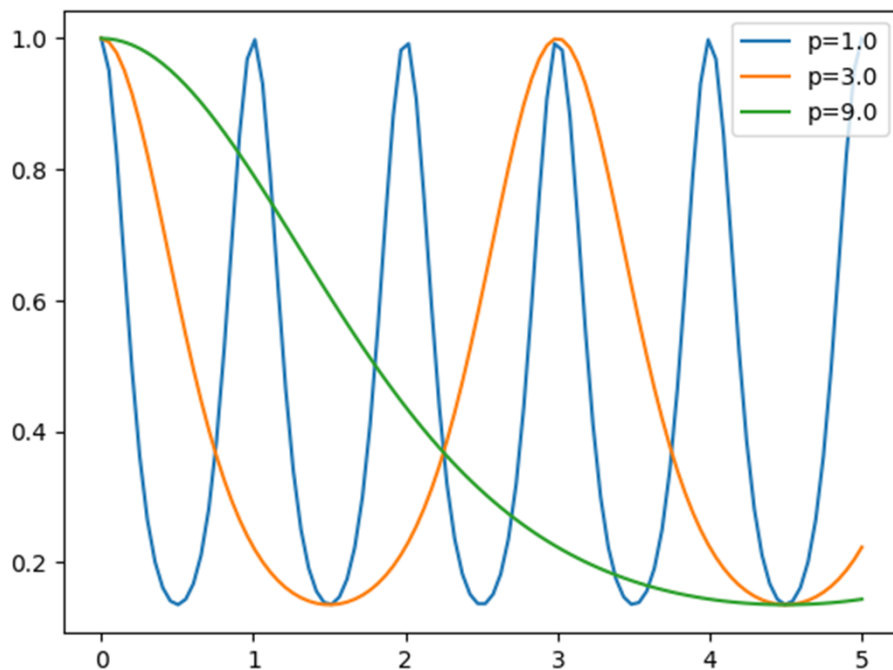




Fonction de Covariance

◉ Il existe plusieurs fonctions de covariance couramment utilisées:

- **Covariance périodique:** $k(t, t') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|t-t'|/p)}{l^2}\right)$

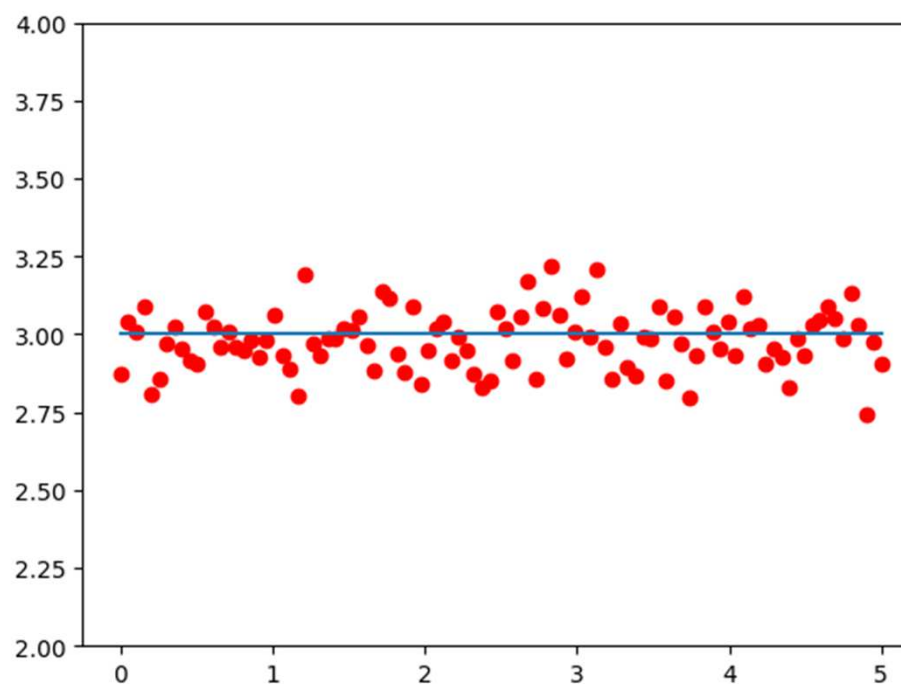




Fonctions Moyenne

◉ Fonctions moyenne souvent utilisées:

- De la forme: $\mathbf{m}(t) = f(t)^T \beta$
- Avec $f(t)$ base de fonctions:
 - Constante: $f(t) = 1$

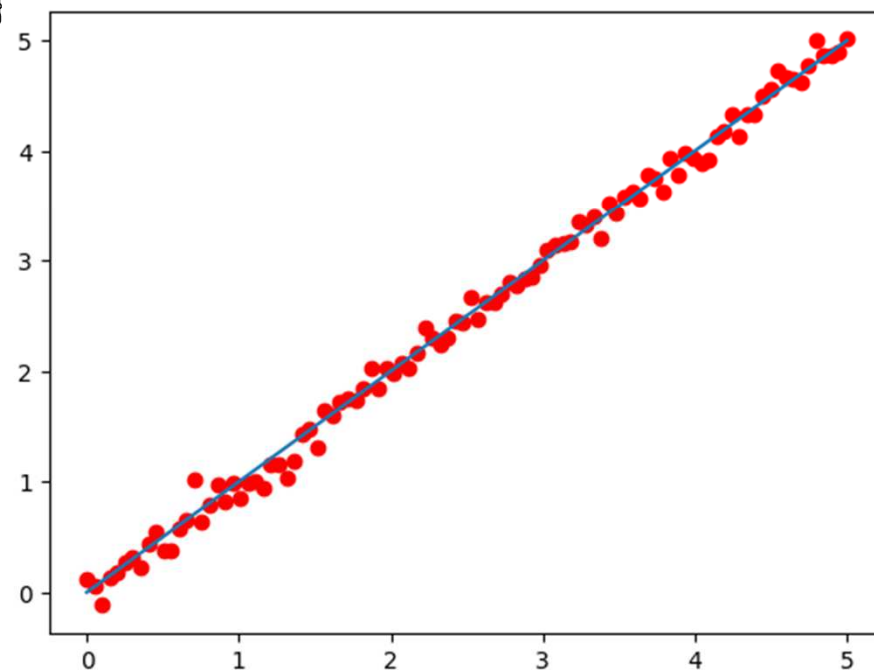
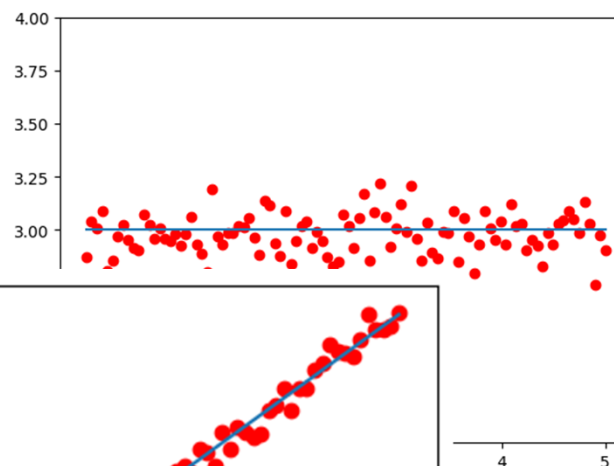




Fonctions Moyenne

○ Fonctions moyenne souvent utilisées:

- De la forme: $\mathbf{m}(t) = f(t)^T \beta$
- Avec $f(t)$ base de fonctions:
 - Constante: $f(t) = 1$
 - Linéaire: $f(t) = \{1, t\}$

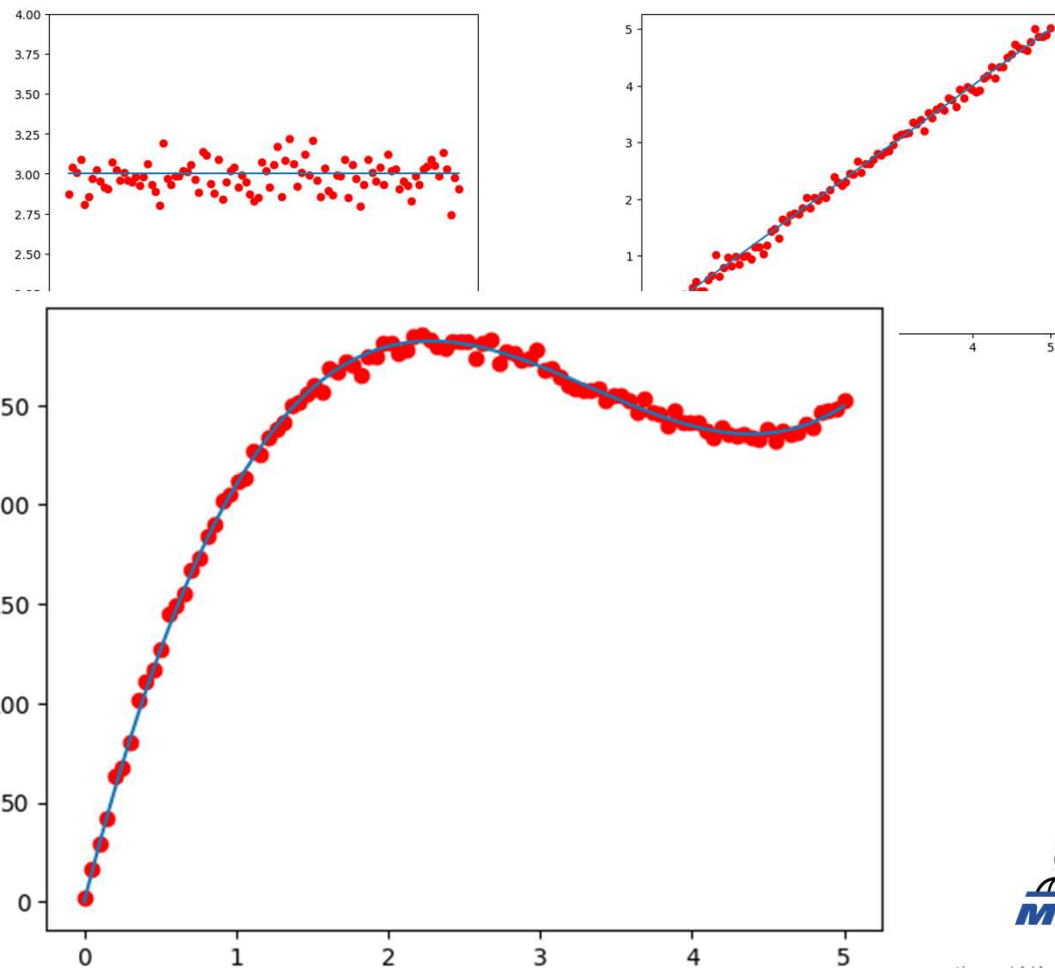




Fonctions Moyenne

○ Fonctions moyenne souvent utilisées:

- De la forme: $\mathbf{m}(\mathbf{t}) = \mathbf{f}(\mathbf{t})^T \boldsymbol{\beta}$
- Avec $\mathbf{f}(\mathbf{t})$ base de fonctions:
 - Constante: $\mathbf{f}(\mathbf{t}) = 1$
 - Linéaire: $\mathbf{f}(\mathbf{t}) = \{1, t\}$
 - Polynomial: $\mathbf{f}(\mathbf{t}) = \{1, t, \dots, t^n\}$





Identification des hyperparamètres

◉ Hyperparamètres à identifier:

- Paramètres de la fonction de covariance: θ (l, p,...)
- Paramètres de la fonction moyenne: β_i
- Valeur de la variance: σ^2

◉ Identification par maximisation de la vraisemblance:

- Base d'apprentissage: $T_s = [t_1, \dots, t_n]^T$, $X_s = [x_1, \dots, x_n]^T$
- Loi jointe: $X_s \sim \mathcal{N}(F_s \beta, K_s)$ avec:

$$F_s = f(T_s) = [f(t_1), \dots, f(t_n)]^T$$

$(K_s)_{i,j}$ la matrice de covariance

- Vraisemblance: $\frac{1}{(2\pi\sigma^2)^{n/2} |K_s|^{1/2}} \exp \left[-\frac{(X_s - F_s \beta)^T K_s^{-1} (X_s - F_s \beta)}{2\sigma^2} \right]$

$$(\hat{\theta}, \hat{\beta}_i, \hat{\sigma}) = \underset{(\theta, \beta_i, \sigma)}{\operatorname{argmax}} \left[-\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2} \log(|K_s|) - \frac{1}{2} \sigma^2 (X_s - F_s \beta)^T K_s^{-1} (X_s - F_s \beta) \right]$$



Identification des hyperparamètres

◉ Expression analytique de β :

- A partir des paramètres de la fonction de covariance θ (l , p, \dots), on peut montrer que:

$$\hat{\beta} = \frac{F_s K_s^{-1} X_s}{F_s^T K_s^{-1} F_s}$$

◉ Expression analytique de σ :

- A partir de $\hat{\beta}$, on a:

$$\hat{\sigma} = \frac{(X_s - F_s \hat{\beta})^T K_s^{-1} (X_s - F_s \hat{\beta})}{n}$$

◉ Identification de θ :

- Substitution de $\hat{\beta}$ et $\hat{\sigma}$ dans $\left[-\frac{n}{2} \log(2\pi\hat{\sigma}^2) - \frac{1}{2} \log(|K_s|) - \frac{1}{2} \hat{\sigma}^2 (X_s - F_s \hat{\beta})^T K_s^{-1} (X_s - F_s \hat{\beta}) \right]$ (*Obj*)
- Estimation de $\hat{\theta}$ en maximisant (*Obj*)



Prédiction par Processus Gaussien

- ◉ Idée: Incorporer la connaissance que les données d'apprentissage peuvent apporter sur la fonction étudiée:

- Selon la distribution à priori donnée par le processus gaussien, la distribution jointe des données d'apprentissage X_s et d'une donnée non observée X^* en t^* est:

$$\begin{pmatrix} X_s \\ X^* \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} F_s \beta \\ f(t^*)^T \beta \end{pmatrix}, \begin{pmatrix} K_s & k(t^*) \\ k(t^*) & 1 \end{pmatrix} \right)$$

$$\text{avec } k(t^*) = [k(t^*, t_i), i = 1, \dots, n]^T$$

- La distribution conditionnelle de X^* sachant X_s est donnée par:

$$X^* | t^*, T_s, X_s \sim \mathcal{N}(\mu(t^*), \tilde{\sigma}^2(t^*))$$

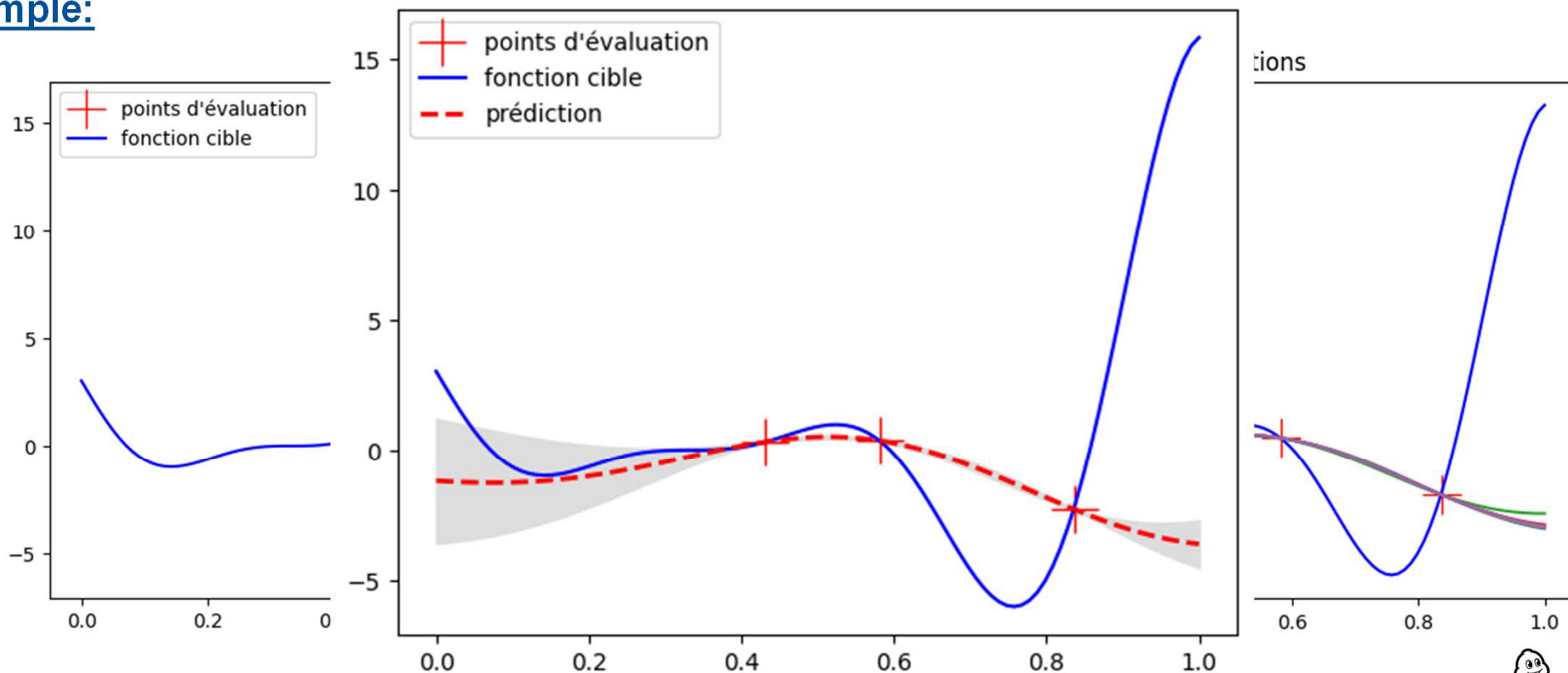
$$\text{Avec } \mu(t^*) = f(t^*)^T \beta + k(t^*) K_s^{-1} [X_s - F_s \beta] \text{ (Prédicteur)}$$

$$\tilde{\sigma}^2(t^*) = \sigma^2 [1 - k(t^*)^T K_s^{-1} k(t^*)] \text{ (Erreur de prédiction)}$$



Prédiction par Processus Gaussien

Exemple:



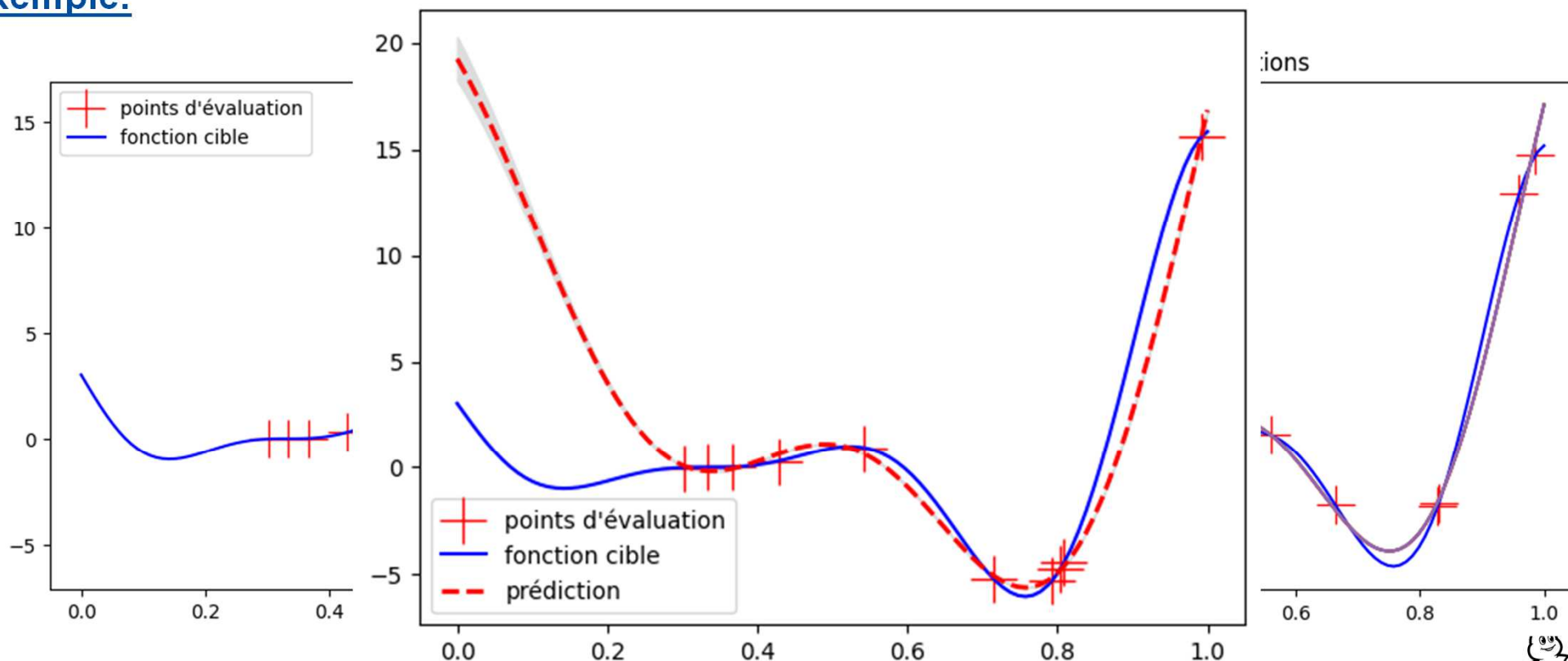
30





Prédiction par Processus Gaussien

Exemple:



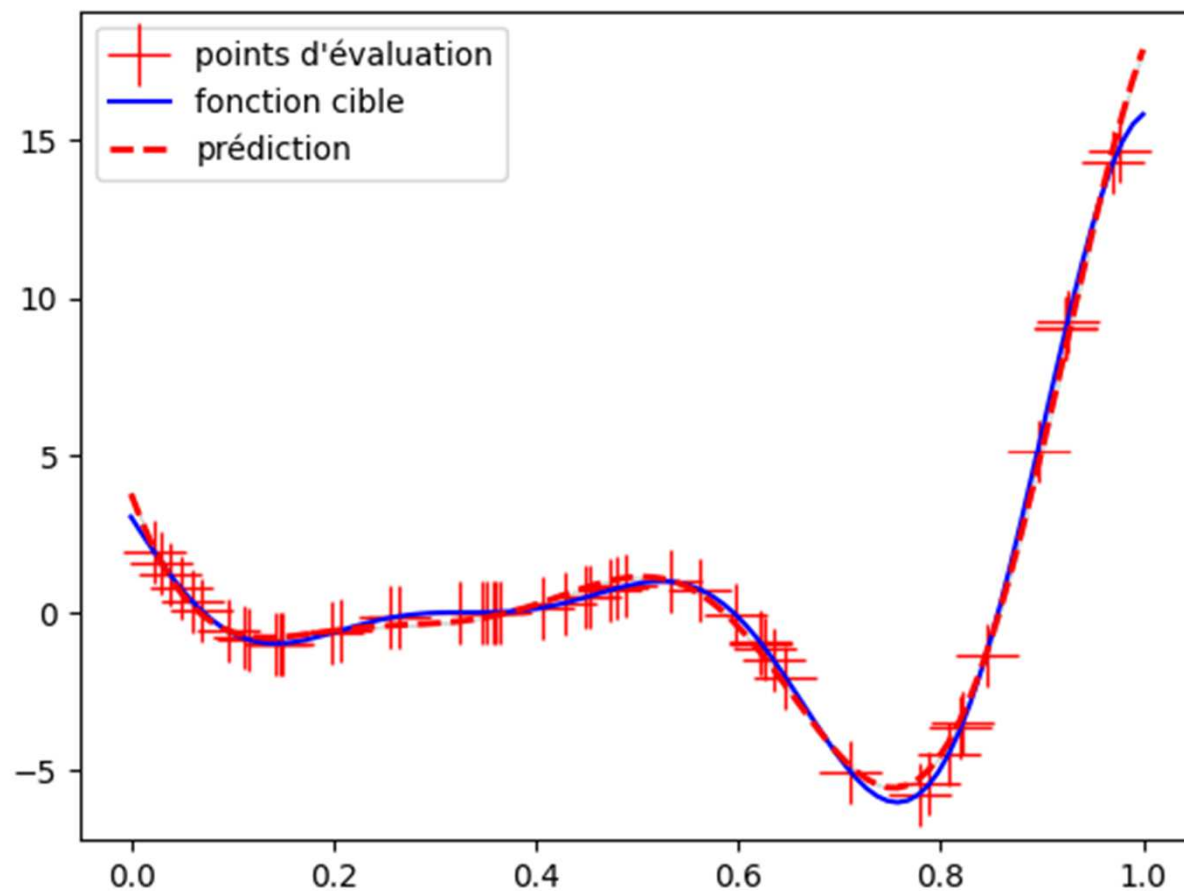
31





Prédiction par Processus Gaussien

◉ Exemple:





Validation du Modèle

Validation croisée:



Apprentissage



Test

Calcul du critère Q_2 :

$$Q_2 = 1 - \frac{\sum_{i=1}^n (X_i - \widehat{X}_i)}{\sum_{i=1}^n (X_i - \bar{X})}$$

- X_i : valeurs observées sur la base de test
- \widehat{X}_i : valeurs prédites par le modèle gaussien sur la base de test
- \bar{X} : valeur moyenne des valeurs observées sur la base de test

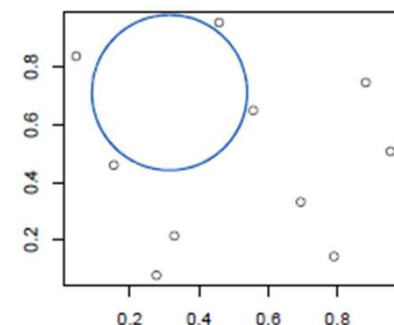


Planification d'Expériences Numériques

- Objectif: Placer des points dans l'espace de recherche de manière à maximiser la quantité d'information que l'on peut avoir sur la fonction boîte noire

- Remplir « uniformément » l'espace:

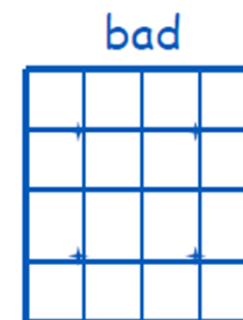
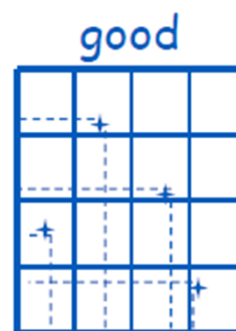
- Utiliser une grille régulière à n niveaux peut s'avérer coûteux quand n est grand
- Faire un échantillonnage de type Monte Carlo est inefficace
- → Utilisation de méthodes de Space Filling



Source: B. Ioos, EDF R&D

- Avoir une bonne répartition des points lorsque l'on réduit la dimension de l'espace

- → Utilisation d'un plan hypercube latin



Source: B. Ioos, EDF R&D



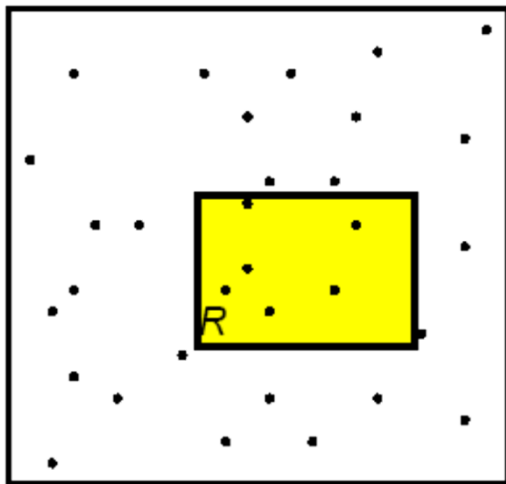
Planification d'Expériences Numériques

◉ Space Filling:

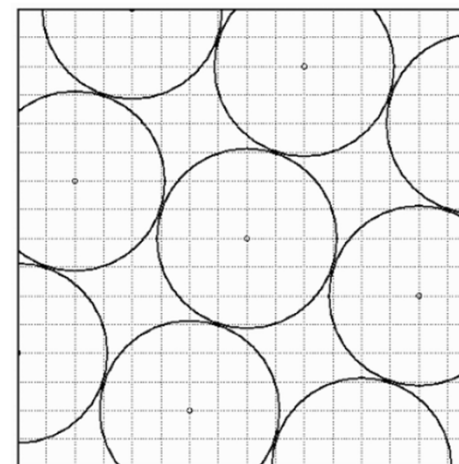
- Critères géométriques (minimax, maximin)
- Critère statistique (faible discrédance)

$$Q(t) = [0, t_1[\times [0, t_2[\times \dots [0, t_p[$$

$$\text{Discrédance} = \sup_{Q(t) \in [0,1]^p} \left| \frac{N_{Q(t)}}{N} - \prod_{i=1}^p t_i \right|$$



Source: B. Ioos, EDF R&D



Source: B. Ioos, EDF R&D

→ Utilisation de suites de Halton, Sobol,...



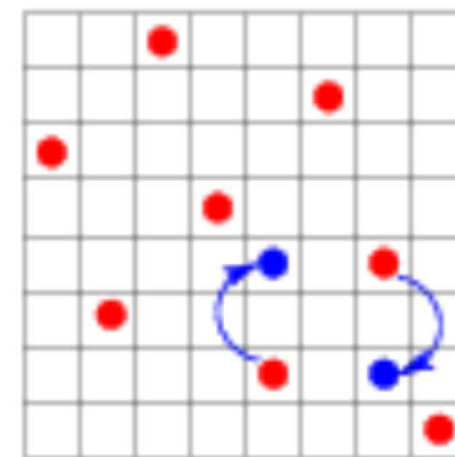
Planification d'Expériences Numériques

◉ LHS optimisé:

- Permet de générer des plans avec:
 - un bon remplissage
 - une répartition uniforme sur les marges

◉ Algorithme (type Recuit Simulé):

- Générer un plan LHS initial et une température T_0
- Tant que $T_k > 0$:
 - Permutation de 2 composantes dans une colonne
 - Accepter le nouveau plan D_{new} avec la probabilité $\min\left(e^{-\frac{\Phi(D_{new}) - \Phi(D)}{T}}, 1\right)$
 - Générer $T_{k+1} < T_k$



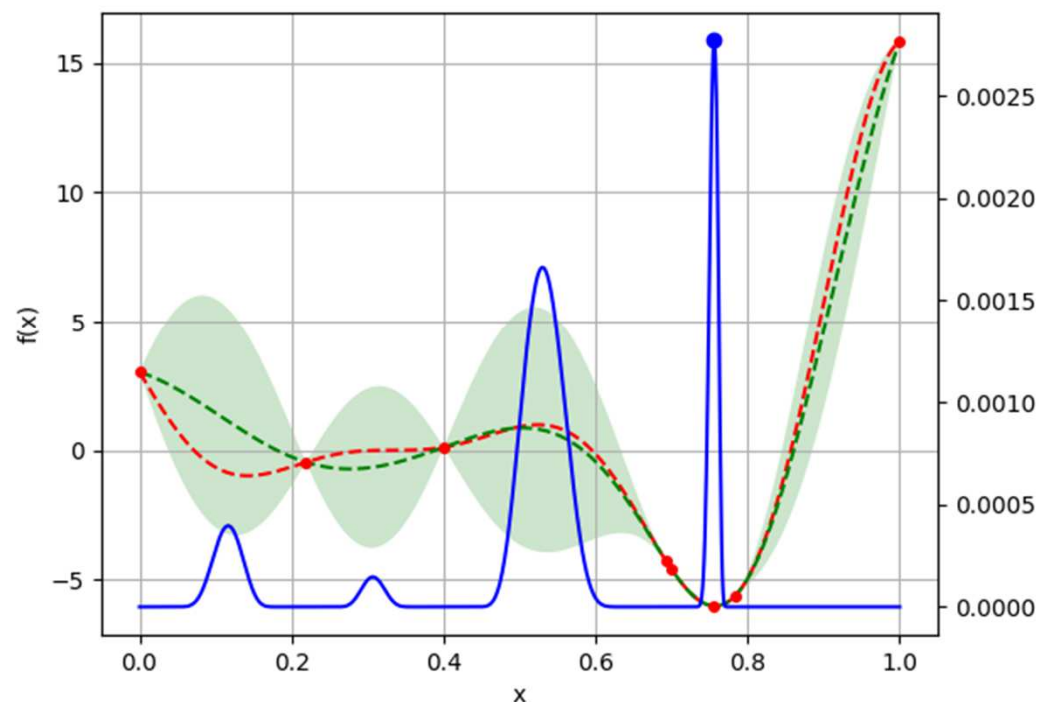
Source: B. Ioos, EDF R&D



Optimisation Bayésienne

Principe:

- Utilisation de la modélisation par processus gaussien comme modèle de substitution d'un modèle type boîte noire ou coûteux en temps de calcul
- Optimisation d'un critère qui s'appuie sur l'exploitation/exploration du processus gaussien

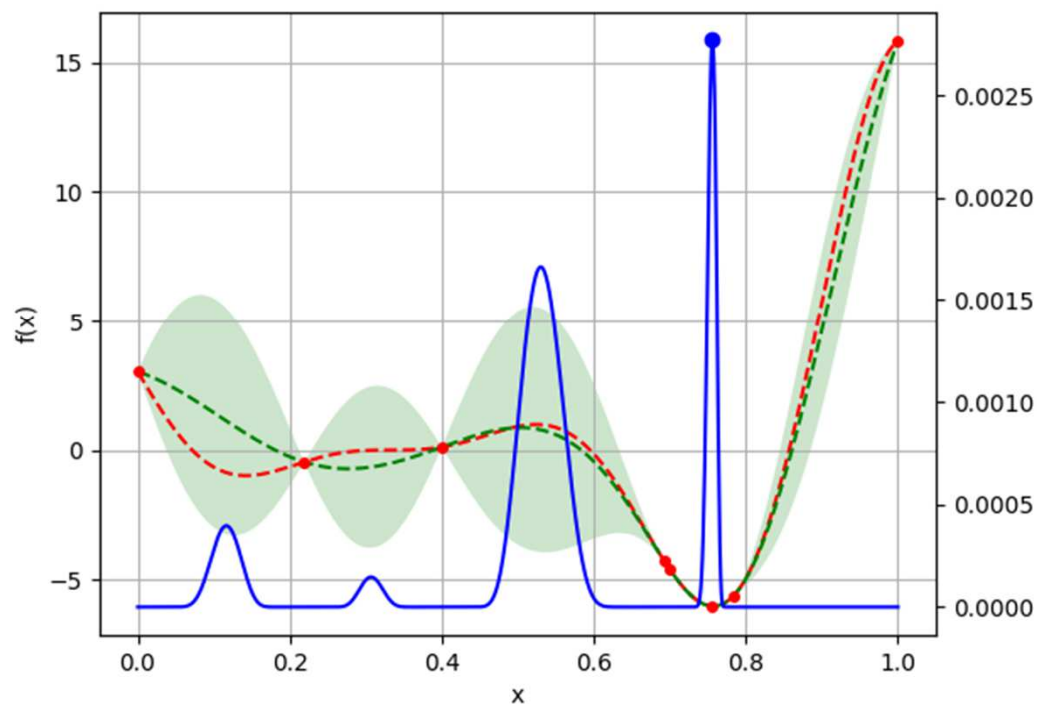




Optimisation Bayésienne

⊙ Algorithme:

- Pour $k = 1, 2, \dots$:
 1. Trouver x_k qui maximise une fonction d'utilité $U(x|x_s)$
 2. Echantillonner la fonction coût en x_k : $y_k = f(x_k)$
 3. Mettre à jour le processus gaussien avec (x_k, y_k)





Dilemme Exploitation/Exploration

- ◉ Rappel de la prédiction par processus gaussien:

$$P(Y^* | x^*, X_s) = \mathcal{N}(\mu(x^*), \tilde{\sigma}^2(x^*))$$

Avec $\mu(x^*) = f(x^*)^T \beta + k(x^*) K_s^{-1} [Y_s - F_s \beta]$ (Prédicteur)
 $\tilde{\sigma}^2(x^*) = \sigma^2 [1 - k(x^*)^T K_s^{-1} k(x^*)]$ (Erreur de prédiction)

- ◉ Le choix du prochain point où l'on doit échantillonner la fonction coût devrait être là où μ est faible et $\tilde{\sigma}^2$ élevée
- ◉ Il est possible de gérer ce compromis en considérant une fonction d'utilité ou d'acquisition comme:

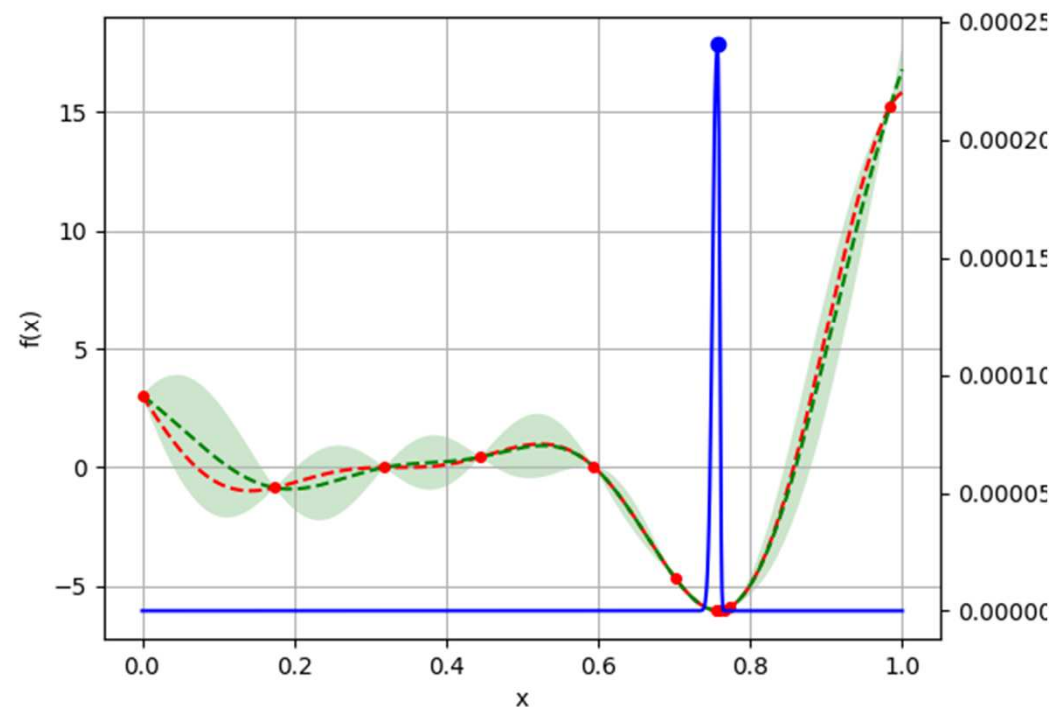
$$-\mu(x^*) + \lambda \tilde{\sigma}^2(x^*)$$



Fonctions d'Acquisition

◉ Rôle d'une fonction d'acquisition:

- Guider l'optimisation en choisissant le prochain point à simuler
- Utilise la prédiction à posteriori pour combiner l'exploitation (régions avec une faible moyenne) et l'exploration (régions avec une forte variance)
- Sert de fonction coût pour trouver les points d'échantillonnage





Probabilité d'Amélioration

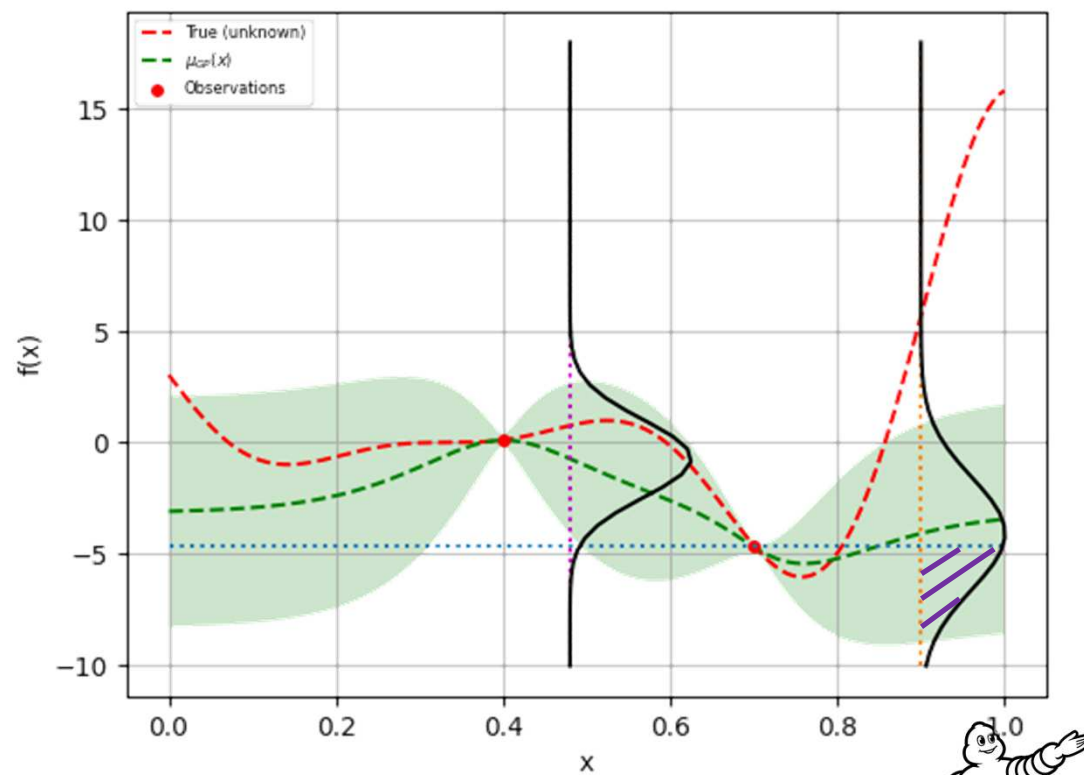
(Probability of Improvement - PI)

◉ Expression de la probabilité d'amélioration:

$$PI(x) = P(f(x) \leq \mu^-)$$

$$PI(x) = \Phi\left(\frac{\mu^- - \mu(x)}{\tilde{\sigma}(x)}\right)$$

Avec Φ la fonction de répartition de $\mathcal{N}(\mu, \sigma^2)$





Amélioration Espérée

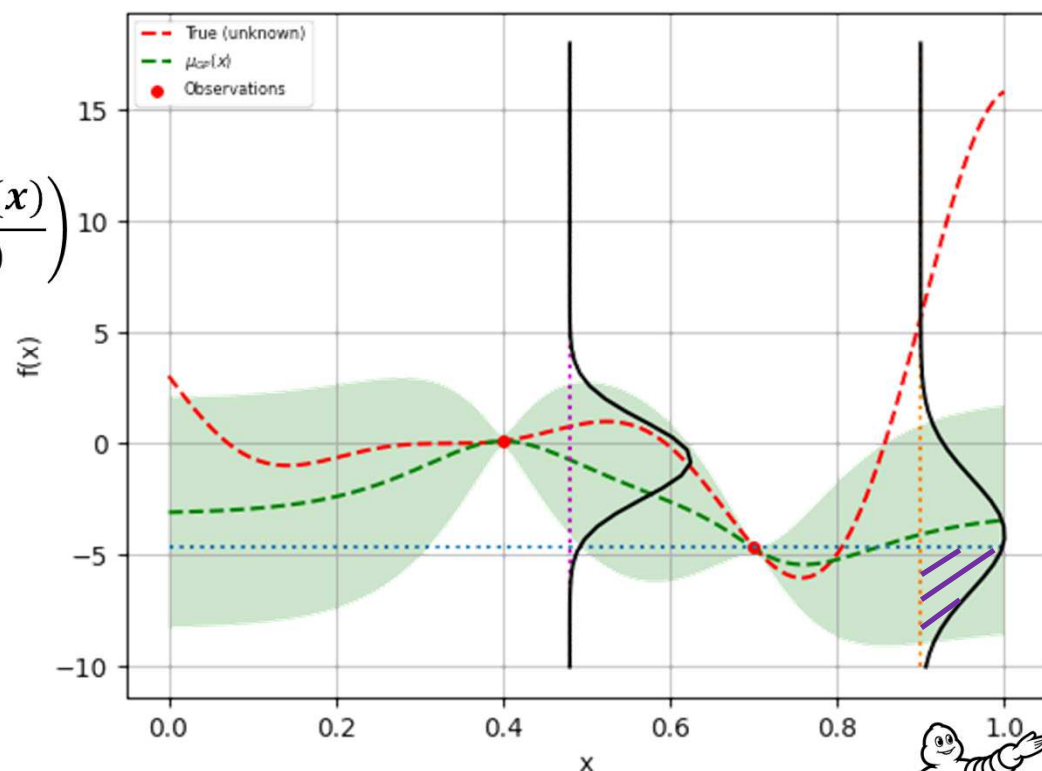
(Expected Improvement- EI)

Expression de l'amélioration espérée:

$$EI(x) = \mathbb{E}[\max(\mu^- - \mu(x), 0)]$$

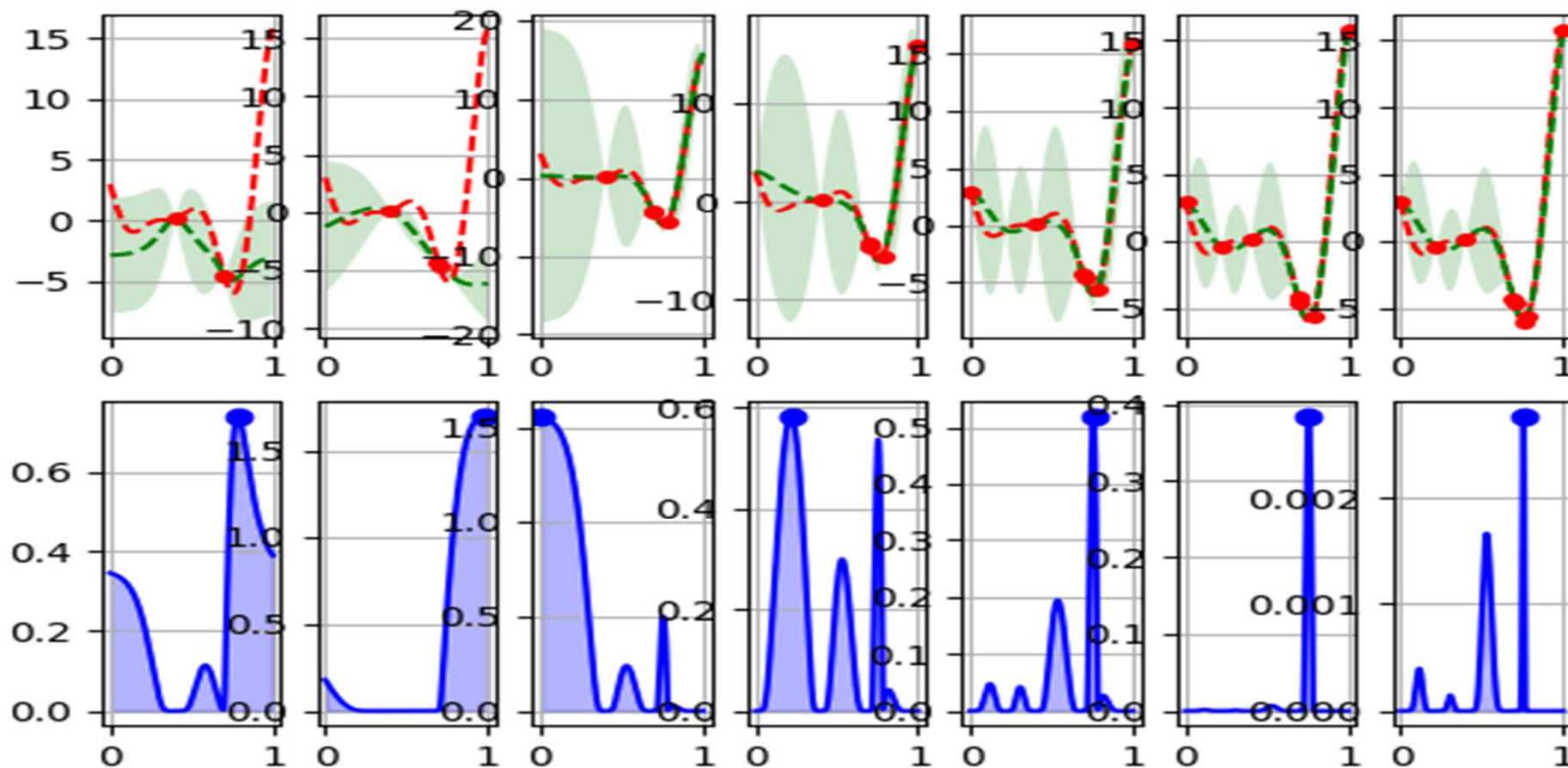
$$EI(x) = (\mu^- - \mu(x))\Phi\left(\frac{\mu^- - \mu(x)}{\tilde{\sigma}(x)}\right) + \tilde{\sigma}^2(x)\phi\left(\frac{\mu^- - \mu(x)}{\tilde{\sigma}(x)}\right)$$

Avec Φ la fonction de répartition et ϕ la densité de probabilité de $\mathcal{N}(\mu, \sigma^2)$



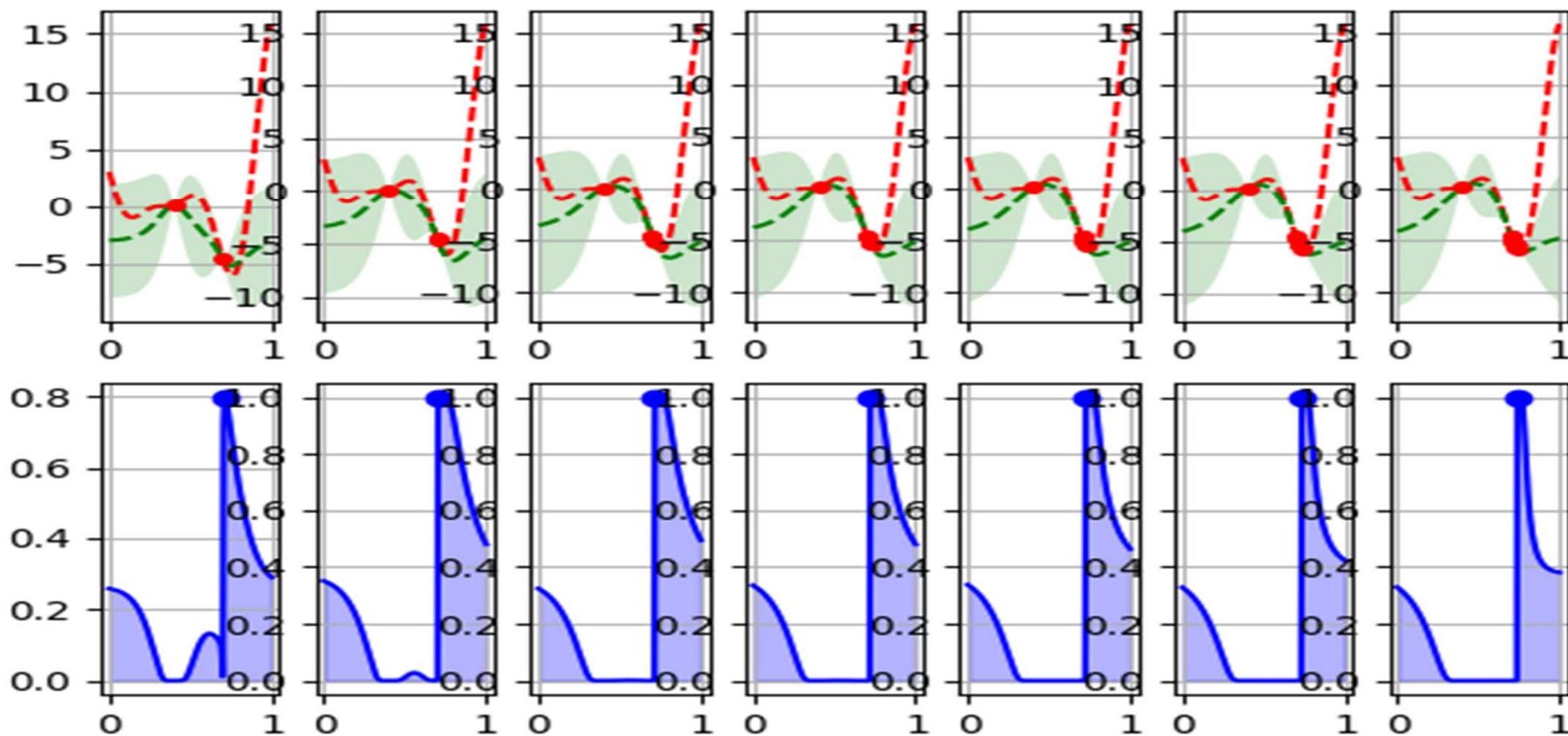


EI





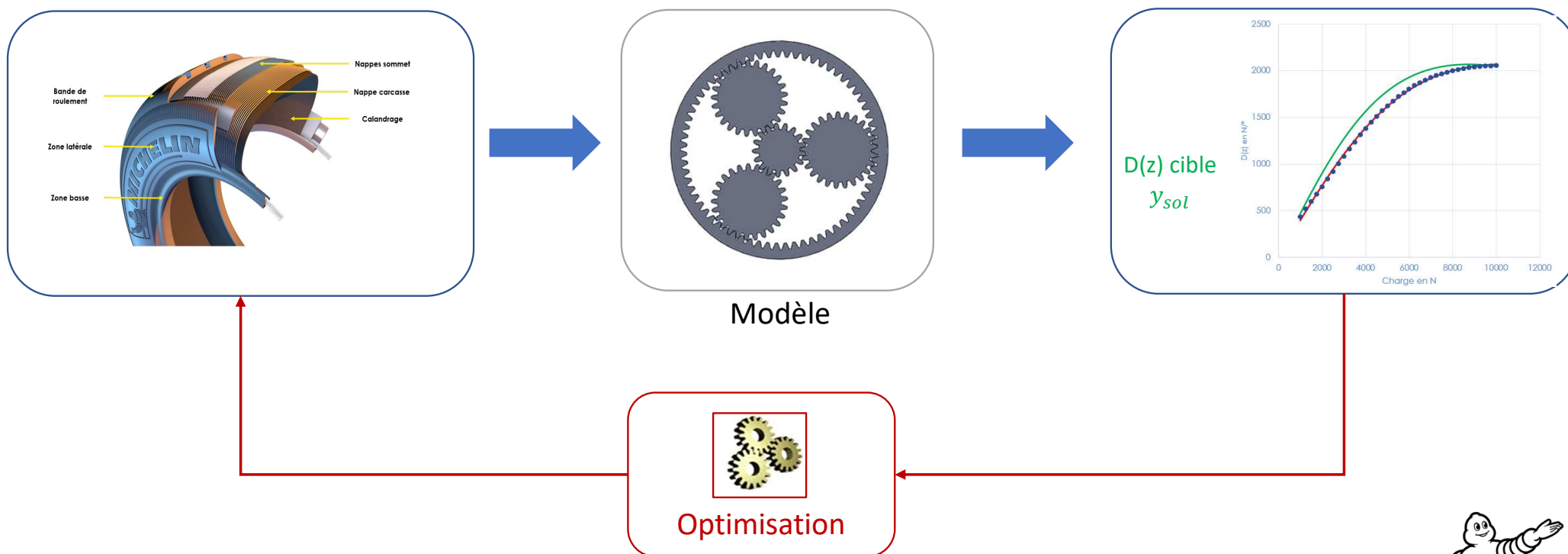
PI





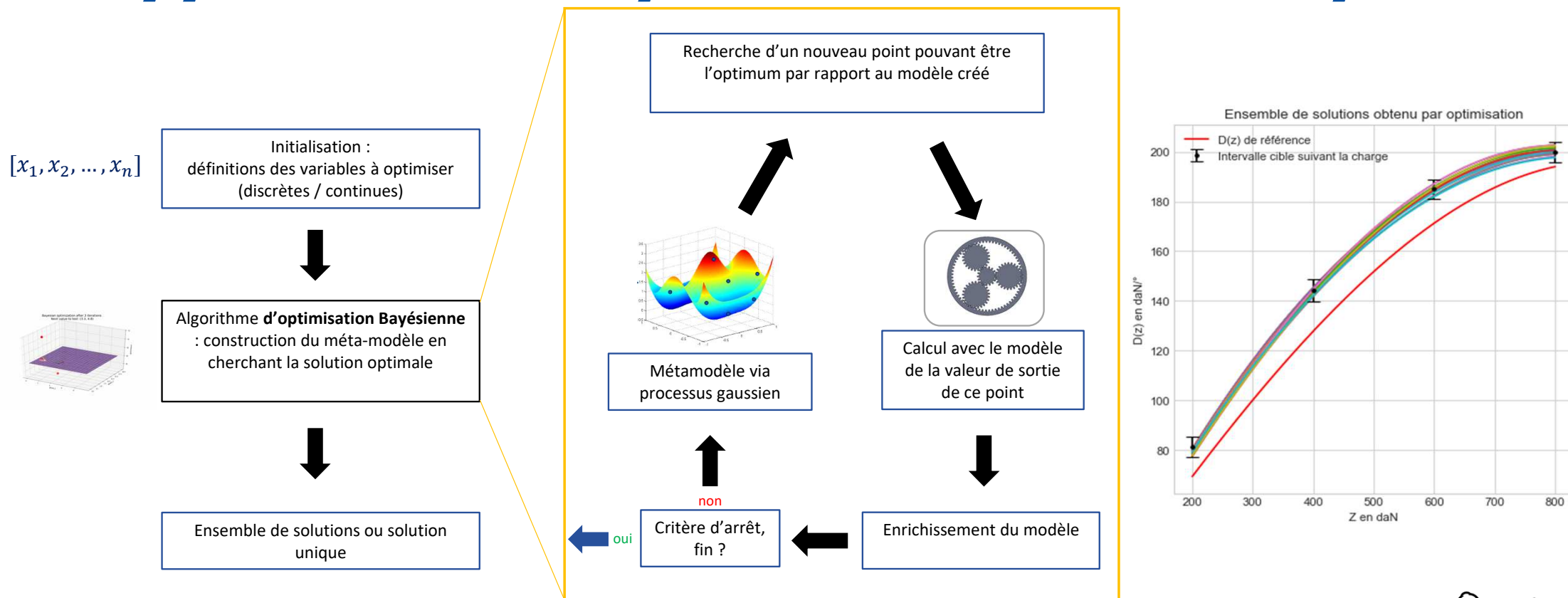
Application en Optimisation de Conception

Optimisation de Conception:





Application en Optimisation de Conception





Conclusion Optimisation Bayésienne

***A vous de vous
exprimer...***



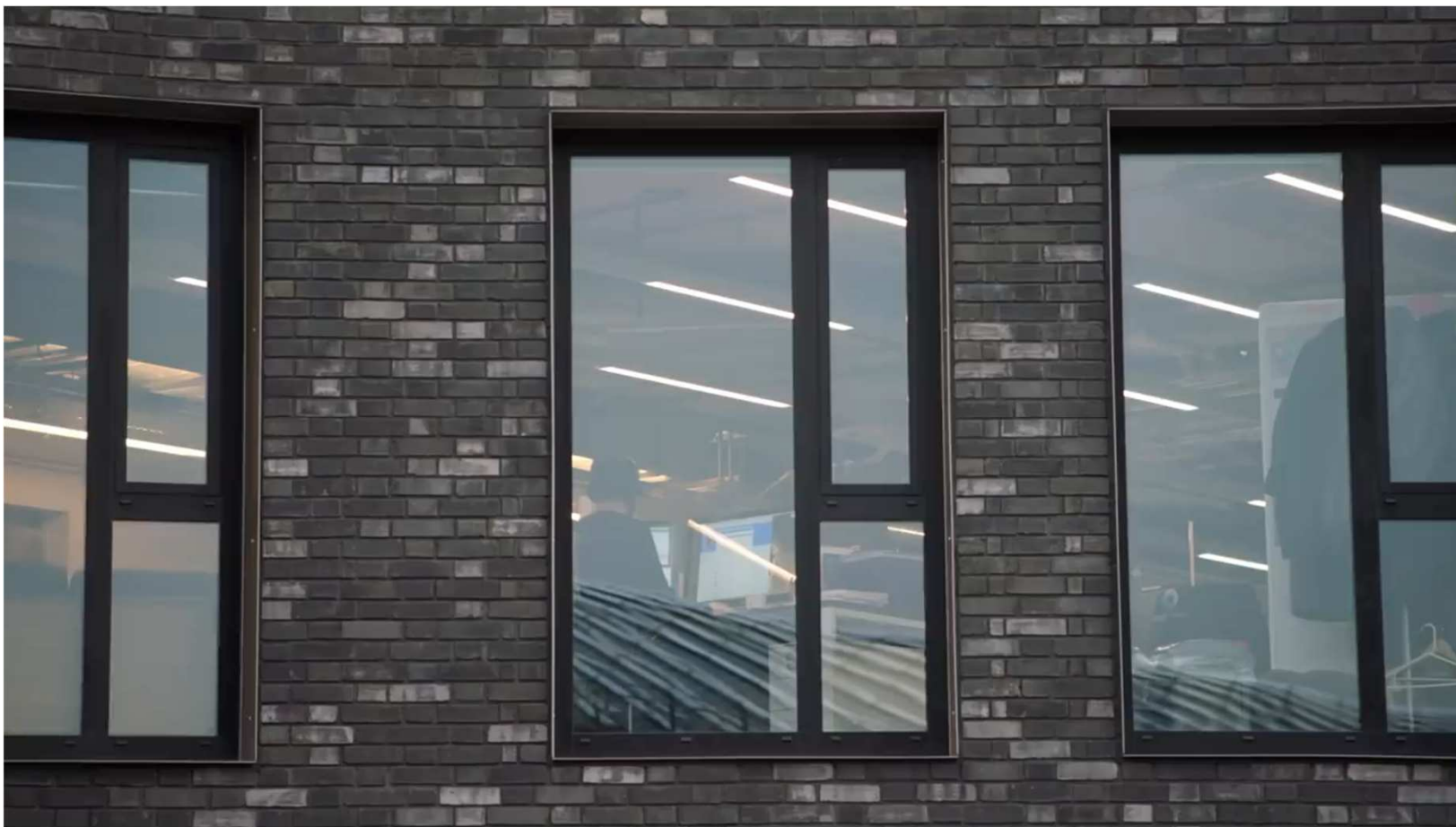
Apprentissage par Renforcement



- 1. Quelques Définitions***
- 2. Apprentissage de la Valeur***
- 3. Apprentissage de la Politique***



Mise en Bouche





Apprentissage par Renforcement

Les différents types d'apprentissage:

Apprentissage Supervisé

Données: (x, y)
 x sont les entrées, y sont les labels

Objectif: Apprendre une fonction: $y = f(x)$



Cet objet est un pneu

Apprentissage Non Supervisé

Données: x
 x sont les données, pas de labels

Objectif: Apprendre une structure sous-jacente



Ces objets sont proches

Apprentissage par Renforcement

Données: couples (état-action)

Objectif: Résoudre un problème



Appliquer cette séquence $(F_x(t), F_y(t))$ d'efforts à ce pneu pour rouler droit



Apprentissage par Renforcement

Les différents types d'apprentissage:

Reinforcement Learning is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them.

R.S.Sutton & A.G.Barto, *Reinforcement Learning*, 1998

Apprentissage par Renforcement
Données: couples (état-action)

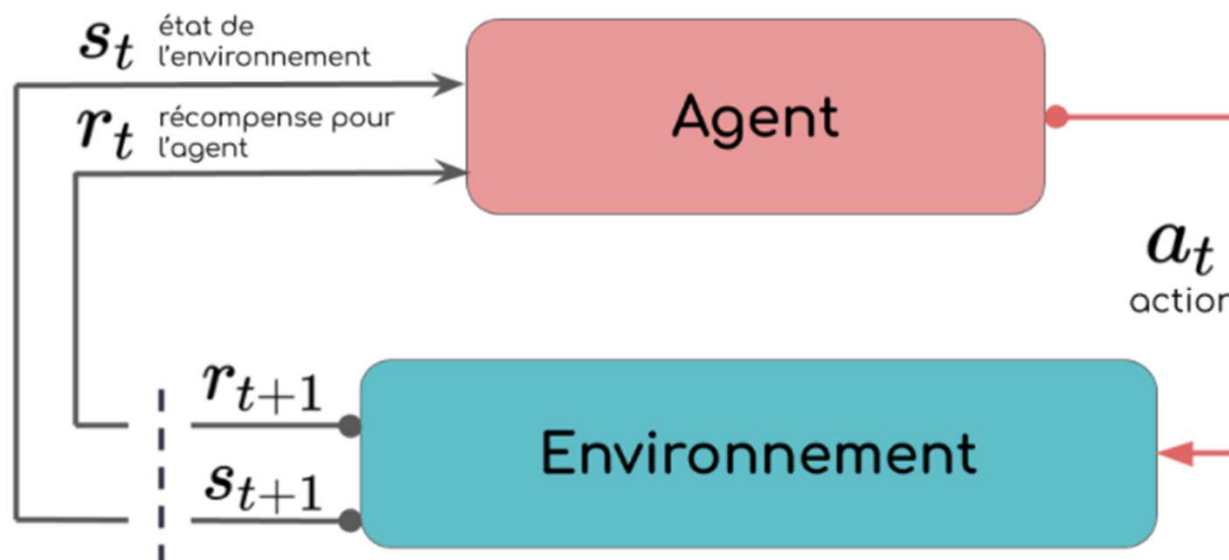
Objectif: Résoudre un problème



Appliquer cette séquence $(F_x(t), F_y(t))$
d'efforts à ce pneu pour rouler droit



Concepts clés



Agent: le programme que l'on entraîne pour accomplir une tâche spécifique

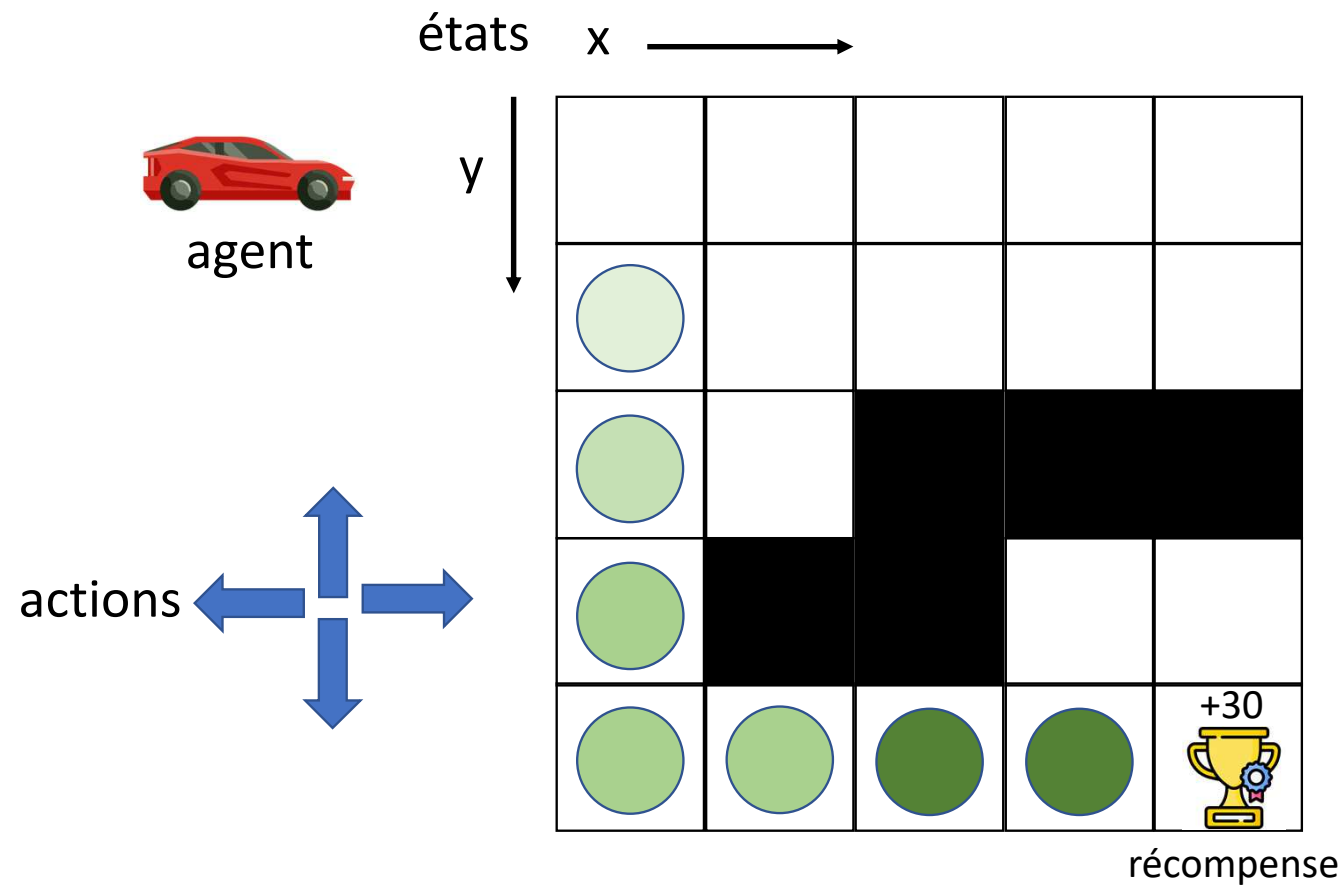
Environnement: le monde, réel ou virtuel, au sein duquel l'agent évolue

Action: une action réalisée par l'agent et qui engendre un changement de l'état de l'environnement

Récompense: l'évaluation de l'impact de l'action qui peut être positive ou négative



Exemple de Cadre pour l'Apprentissage par Renforcement





Processus de Markov

⊙ Propriété de Markov:

- Un processus stochastique présente la propriété de Markov lorsque:

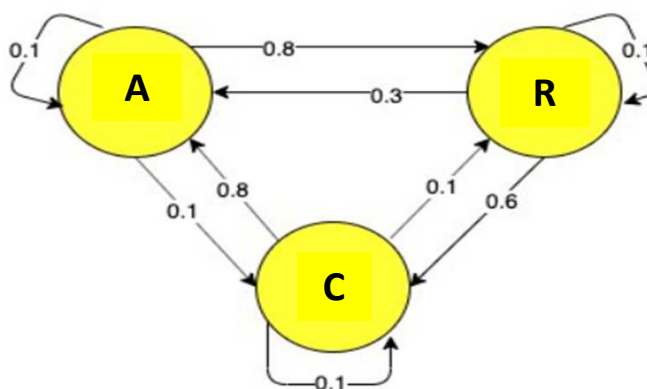
$$P(X(t+1) = j | X(0) = i_0, X(1) = i_1, \dots, X(t) = i) = P(X(t+1) = j | X(t) = i)$$

⊙ Chaîne de Markov

- Modèle stochastique décrivant une séquence d'événements possibles dans laquelle la probabilité de chaque événement dépend uniquement de l'événement précédent
- Elle est notée (S, P) avec:
 - S : ensemble des états
 - $P(s, s')$: probabilité de transition entre l'état s et s'



Exemple de Chaine de Markov



$$P(A) = 40\%$$

$$P(C) = 30\%$$

$$P(R) = 30\%$$

$$P(ARC) = P(x_0 = A)P(x_1 = R|x_0 = A)P(x_2 = C|x_1 = R) = 0,4 * 0,8 * 0,6 = 0,192$$

$$P(RCA) = P(x_0 = R)P(x_1 = C|x_0 = R)P(x_2 = A|x_1 = C) = 0,3 * 0,6 * 0,8 = 0,144$$

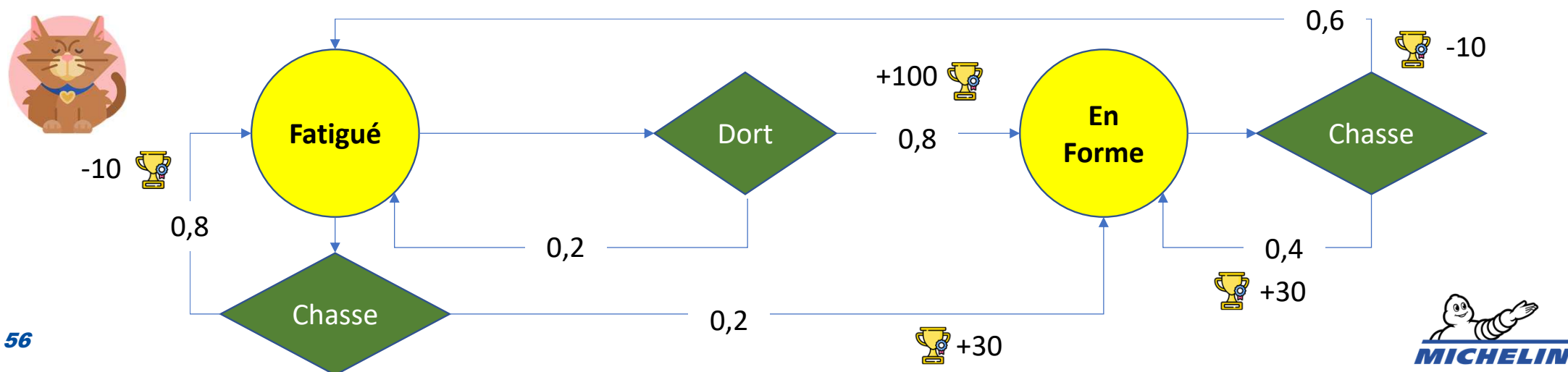
$$P(CAR) = P(x_0 = C)P(x_1 = A|x_0 = C)P(x_2 = R|x_1 = A) = 0,3 * 0,8 * 0,8 = 0,192$$



Processus de Décision Markovien

○ Définition:

- Un processus de décision markovien (PDM) est un 5-uplet (S, A, P, R, γ) avec:
 - S : ensemble des états
 - A : ensemble des actions
 - $P(s, a, s')$: probabilité que l'action a à l'état s amène vers l'état s'
 - $R(s, a, s')$: récompense reçu immédiatement après la transition de l'état s vers l'état s'
 - γ : facteur d'escompte utilisé pour le calcul d'une récompense à rabais





Gain Cumulé et Politique

⊙ Fonction de récompense:

$$r_t = R(s_t, a_t)$$

⊙ Objectif de l'agent: Maximiser un gain cumulé

- Gain cumulé sur un horizon fini:

$$R(\tau) = \sum_{t=0}^T r_t$$

Avec $\tau = (s_0, s_1, \dots, s_T)$

- Gain cumulé sur un horizon infini:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

Avec $\tau = (s_0, s_1, \dots)$ et $\gamma \in (0, 1)$

⊙ Politique:

- Déterministe:

$$a_t = \mu(s_t)$$

- Stochastique:

$$a_t \sim \pi(\cdot | s_t)$$



Fonctions de Valeurs

- ◉ Valeur d'un état:

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi}[R(\tau) | s_0 = s]$$

- ◉ Valeur d'un couple état/action:

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi}[R(\tau) | s_0 = s, a_0 = a]$$

- ◉ Valeur optimale d'un état:

$$V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi}[R(\tau) | s_0 = s]$$

- ◉ Valeur optimale d'un couple état/action:

$$Q^*(s, a) = \max_{\pi} \mathbb{E}_{\tau \sim \pi}[R(\tau) | s_0 = s, a_0 = a]$$



Approches d'Apprentissage par Renforcement

Apprentissage de la Valeur

Trouver $Q(s, a)$

Déterminer:
 $a^* = \operatorname{argmax}_a Q(s, a)$

Apprentissage de la Politique

Trouver $\pi(s)$

Tirage aléatoire selon:
 $a \sim \pi(s)$



Approches d'Apprentissage par Renforcement

Apprentissage de la Valeur

Trouver $Q(s, a)$

Déterminer:
 $a^* = \operatorname{argmax}_a Q(s, a)$



Equations de Bellman

- ◉ Idée:
 - La valeur de l'état de départ est égale à la récompense immédiate plus la valeur de l'état futur

- ◉ Equations de Bellman pour les fonctions de valeur:

$$V^{\pi}(s) = \mathbb{E}_{\substack{a \sim \pi \\ s' \sim P}} [r(s, a) + \gamma V^{\pi}(s')]]$$

$$Q^{\pi}(s, a) = \mathbb{E}_{s' \sim P} [r(s, a) + \gamma \mathbb{E}_{a' \sim \pi} [Q^{\pi}(s', a')]]$$

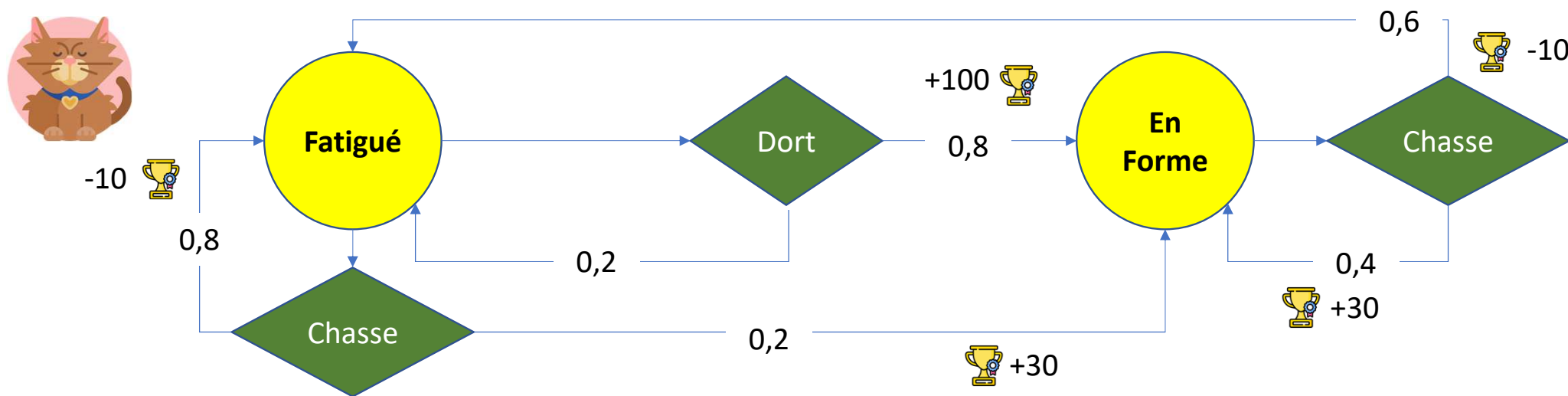
- ◉ Equations de Bellman pour les fonctions de valeur optimales:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')]]$$

$$Q^*(s, a) = \mathbb{E}_{s' \sim P} [r(s, a) + \gamma \max_{a'} Q^*(s', a')]]$$



Application Equation de Bellman



	A_0	A_1
S_0	390.90	340.39
S_1	-inf	337.90



Estimation des Fonctions de Valeur

◉ Comment faire lorsque le modèle est inconnu?

- → Tenter de trouver les transitions de probabilité P
- → L'agent doit apprendre de son environnement en interagissant avec lui et collecter des expériences

◉ Approche Monte-Carlo:

- Réaliser beaucoup d'essais consistant à exécuter une série d'actions et à récupérer les récompenses correspondantes
- Chaque essai est appelé épisode et se termine lorsque l'on atteint l'état final du processus de décision markovien (Processus de Décision Markovien Episodique)

$$G_t = r_{t+1} + \gamma r_{t+2} + \dots + \gamma^{T-1} r_T$$

- Mise à jour de la fonction valeur:

$$V(s_t) = V(s_t) + \alpha [G_t - V(s_t)]$$

Avec $\alpha \in [0, 1]$



Estimation des Fonctions de Valeur

◉ Comment faire lorsque le modèle est inconnu?

- → Tenter de trouver les transitions de probabilité P
- → L'agent doit apprendre de son environnement en interagissant avec lui et collecter des expériences

◉ Approche par Différence Temporelle:

- Permet d'éviter de mettre à jour $V(s_t)$ uniquement à la fin de l'épisode
- Met à jour $V(s_t)$ juste après avoir réalisé l'action a_t
- Mise à jour de la fonction valeur:

$$V(s_t) = V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

Où $r_{t+1} + \gamma V(s_{t+1})$ est appelé valeur de la cible différence temporelle

$r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ est appelé erreur de la différence temporelle



Q-Learning

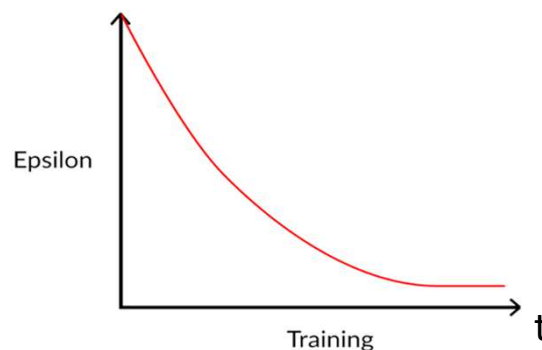
◉ Idée:

- Exploiter la stratégie de différence temporelle en appliquant une politique équilibrant exploitation/exploration
- Mise à jour:

$$Q(s_t, a) = Q(s_t, a) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a) \right]$$

◉ Politique d'exploration ϵ -greedy:

- Permet de gérer le compromis exploitation/exploration
 - Avec une probabilité $1 - \epsilon$: on fait de l'exploitation (l'agent sélectionne l'action qui donne le $Q(s_{t+1}, a)$ la plus élevée)
 - Avec une probabilité ϵ : on fait de l'exploration (l'agent sélectionne une action au hasard)







Algorithme Q-Learning

◉ Algorithme:

- Initialiser $Q(s, a) = 0$ pour toutes les paires (s, a)
- Pour $k = 1, \dots, n_{\text{episodes}}$:
 - $\epsilon = \epsilon_k$
 - $t = 0$
 - Tant que $S_t \neq S_{\text{Terminal}}$:
 - Choisir a_t selon la politique ϵ -greedy
 - Appliquer a_t puis collecter r_{t+1} et S_{t+1}
 - $Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$
 - $t = t + 1$
- Retourner Q



Exemple Q-Learning

S_0	S_1	S_2
-1 	-1	+10 



Episode	Instant	Etat	Q(s,a)				Action	R(s,a)	Nouveau Q(s,a)			
1	0	S ₀		S ₀	S ₁	S ₂	→	-1		S ₀	S ₁	S ₂
			←	0	0	0			←	0	0	0
			→	0	0	0			→	-1	0	0
1	1	S ₁		S ₀	S ₁	S ₂	→	+10		S ₀	S ₁	S ₂
			←	0	0	0			←	0	0	0
			→	-1	0	0			→	-1	10	0

$$0 + 1 \times (-1 + 0,9 \times 0 - 0) = -1$$

$$0 + 1 \times (10 + 0,9 \times 0 - 0) = 10$$



Exemple Q-Learning

S_0	S_1	S_2
-1 	-1	+10 

Episode	Instant	Etat	Q(s,a)				Action	R(s,a)	Nouveau Q(s,a)			
2	0	S ₀		S ₀	S ₁	S ₂	←	-1		S ₀	S ₁	S ₂
			←	0	0	0			←	-1	0	0
			→	-1	10	0			→	-1	10	0
2	1	S ₀		S ₀	S ₁	S ₂	→	-1		S ₀	S ₁	S ₂
			←	-1	0	0			←	-1	0	0
			→	-1	10	0			→	8	10	0

$$0 + 1 \times (-1 + 0,9 \times 0 - 0) = -1$$

$$-1 + 1 \times (-1 + 0,9 \times 10 - (-1)) = 8$$





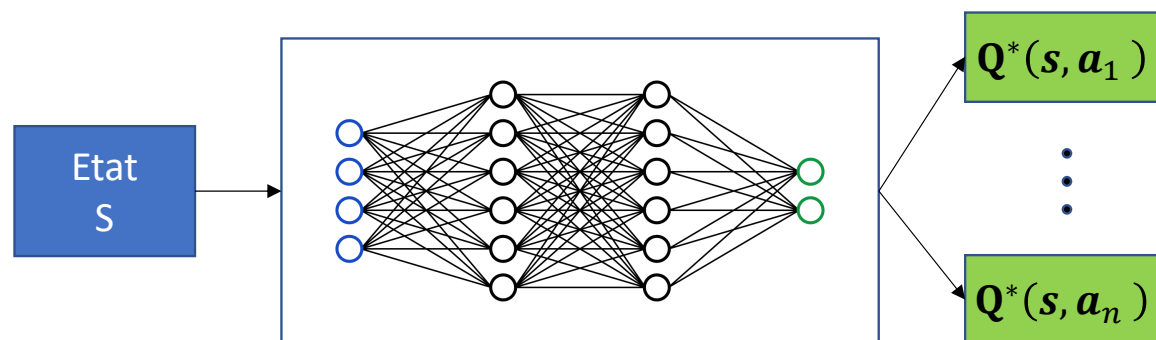
Deep Q-Learning

◉ Pourquoi du Deep Learning?

- L'approche Q-Learning est adaptée lorsque les états et les action sont discrets et de faible dimension
- En grande dimension, l'utilisation d'une table $Q(s, a)$ n'est pas pertinente
- → Mise en place d'une fonction permettant de prédire les $Q^*(s, a)$

◉ Utilisation de réseaux de neurones profonds:

- Efficaces pour traiter des tables $Q(s, a)$ de grande dimension avec des états continus
- La possibilité de combiner des réseaux de neurones convolutionnels et du Q-Learning a permis d'étendre les champs d'application de l'apprentissage par renforcement (exemple: l'état peut être une image)





Stratégie de l'Experience Replay

◉ Objectifs:

- **Eviter d'oublier des expériences passées**

- Les expériences ne sont pas sauvegardées au fil de l'eau → si l'agent rencontre de nouveau une expérience déjà vécue, il risque de ne pas adopter le comportement adéquat

- **Réduire les corrélations entre les expériences**

- Les séquences des états successifs donnés par l'environnement sont souvent corrélés → l'agent risque d'être influencé par cette corrélation

◉ Principe:

- **Créer un buffer qui stocke les expériences au fil des itérations**

- **Prendre aléatoirement des expériences pour les présenter au réseau de neurones**



Algorithme Deep Q-Learning

◉ Algorithme:

- Initialiser un buffer D de taille N
- Initialiser la fonction Q à 0
- Pour $k = 1, \dots, \text{nepisodes}$:
 - Pour $t = 0, \dots, T$:
 - Choisir l'action a_t avec une probabilité ϵ
 - Sinon choisir $a_t = \max_a Q^*(s_t, a_t; \theta)$
 - Exécuter l'action a_t puis collecter r_t et s_{t+1}
 - Stocker l'expérience (s_t, a_t, r_t, s_{t+1}) dans D
 - Prendre aléatoirement un minibatch d'expériences (s_j, a_j, r_j, s_{j+1}) dans D
 - $y_j = r_j + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)$
 - Appliquer une descente de gradient sur $\left(y_j - Q(s_j, a_j; \theta)\right)^2$



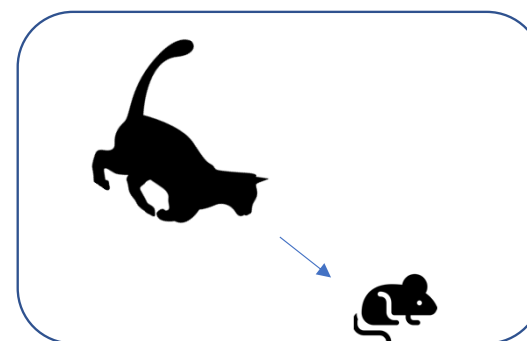
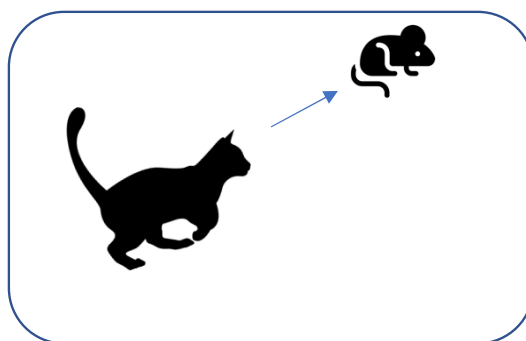
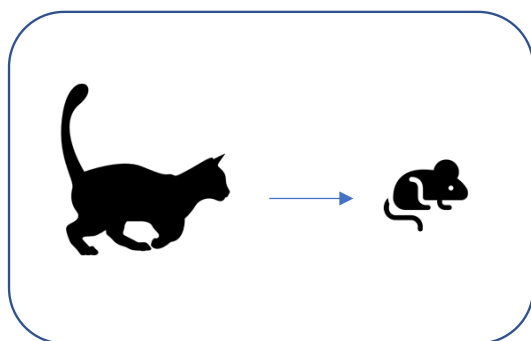
Gestion des Cibles Mouvantes

◉ Rappel de l'erreur TD:

$$TD_{\epsilon} = (r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

◉ Problème:

- Les mêmes poids sont utilisés pour la fonction la valeur courante Q et la cible
- A chaque itération, ces deux grandeurs vont changer → Oscillations dans la convergence





Algorithme

◉ Stratégie de Résolution (proposée par DeepMind):

- Utiliser un réseau de neurones séparé (appelé réseau cible) avec des paramètres fixés
- A chaque itération multiple de κ , copie des paramètres du réseau DQN vers le réseau cible

$$\Delta w = \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a, w^-) - Q(s_{t+1}, a, w) \right] \nabla_w Q(s_t, a, w)$$

$w^- = w$ à chaque pas multiple de κ

◉ Implémentation:

1. Créer deux réseaux (DQN et Cible)
2. Calculer la cible TD avec le réseau Cible
3. Mettre à jour les poids du réseau Cible à chaque " κ -pas"



Approches d'Apprentissage par Renforcement

Apprentissage de la Valeur

Trouver $Q(s, a)$

Déterminer:
 $a^* = \operatorname{argmax}_a Q(s, a)$

Apprentissage de la Politique

Trouver $\pi(s)$

Tirage aléatoire selon:
 $a \sim \pi(s)$



Policy Gradient

◉ Principe:

- Apprendre directement une politique π_{θ} qui relie un état à une action (choisir une action sans utiliser une fonction de valeur)
 - Fonction coût: $J(\theta)$
 - Gradient: $\nabla_{\theta}J(\theta)$
 - Mise à jour: $\theta_{k+1} = \theta_k + \alpha \nabla_{\theta}J(\theta)$

◉ Avantages:

- Convergence:
 - Meilleure convergence (convergence locale) que les approches basées sur l'apprentissage de la fonction valeur (oscillations dues à des changements des valeurs des actions)
- Possibilité d'apprendre des politiques stochastiques:
 - Prise en compte nativement de l'exploration de l'environnement

◉ Inconvénient:

- Convergence locale
- Convergence lente

75





Policy Gradient

◉ Éléments utiles pour le calcul du gradient:

- On considère une politique stochastique π_θ
- Probabilité d'une trajectoire:

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t)$$

- Astuce de la dérivée du \log :

$$\nabla_\theta P(\tau|\theta) = P(\tau|\theta) \nabla_\theta \log P(\tau|\theta)$$

- Log probabilité d'une trajectoire:

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^T (\log P(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t))$$

- Gradient du log de la probabilité d'une trajectoire:

$$\begin{aligned} \nabla_\theta \log P(\tau|\theta) &= \cancel{\nabla_\theta \log \rho_0(s_0)} + \sum_{t=0}^T (\cancel{\nabla_\theta \log P(s_{t+1}|s_t, a_t)} + \nabla_\theta \log \pi_\theta(a_t|s_t)) \\ \nabla_\theta \log P(\tau|\theta) &= \sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t|s_t) \end{aligned}$$



◉ Calcul du gradient:

Policy Gradient

$$\begin{aligned}
 \nabla_{\theta} J(\pi_{\theta}) &= \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)] \\
 &= \nabla_{\theta} \int P(\tau|\theta) R(\tau) \\
 &= \int \nabla_{\theta} P(\tau|\theta) R(\tau) \\
 &= \int P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau) \\
 &= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)]
 \end{aligned}$$

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right] \approx \frac{1}{|D|} \sum_{\tau \in D} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau)$$

Avec D un ensemble de N trajectoires



Approche Acteur-Critique

◉ Problème de la méthode du Policy Gradient:

- Approche de Monte-Carlo → On attend la fin d'un épisode pour avoir le gain cumulé
- Risque d'avoir des grandes différences de gains cumulés entre différentes trajectoires → Risque de convergence lente car gradient bruité, nécessité d'avoir beaucoup d'exemples



◉ Idée: faire une mise à jour à chaque pas de temps



Approche Acteur-Critique

○ Principe:

- Mise à jour à chaque pas de temps

$$\begin{aligned}\Delta\theta &= \alpha \nabla_{\theta} \log \pi_{\theta}(s_t|a_t) R(t) \\ \Delta\theta &= \alpha \nabla_{\theta} \log \pi_{\theta}(s_t|a_t) Q(s_t, a_t)\end{aligned}$$

- Utilisation de deux réseaux de neurones en parallèle:

- Un réseau estime $\widehat{Q}_{\phi}(s_t, a_t)$ (critique)
- Un réseau estime la politique $\pi_{\theta}(s_t|a_t)$ (acteur)

- Mises à jour:

- Acteur:

$$\Delta\theta = \alpha \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) Q_{\phi}(s_t, a_t)$$

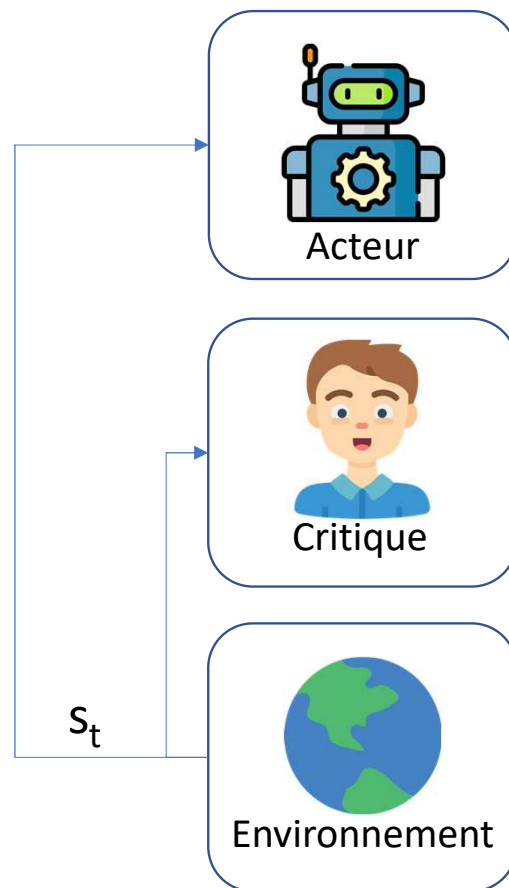
- Critique:

$$\Delta\phi = \beta (R(s_t, a_t) + \gamma \widehat{Q}_{\phi}(s_{t+1}, a_{t+1}) - \widehat{Q}_{\phi}(s_t, a_t)) \nabla_{\phi} \widehat{Q}_{\phi}(s_t, a_t)$$



Approche Acteur-Critique

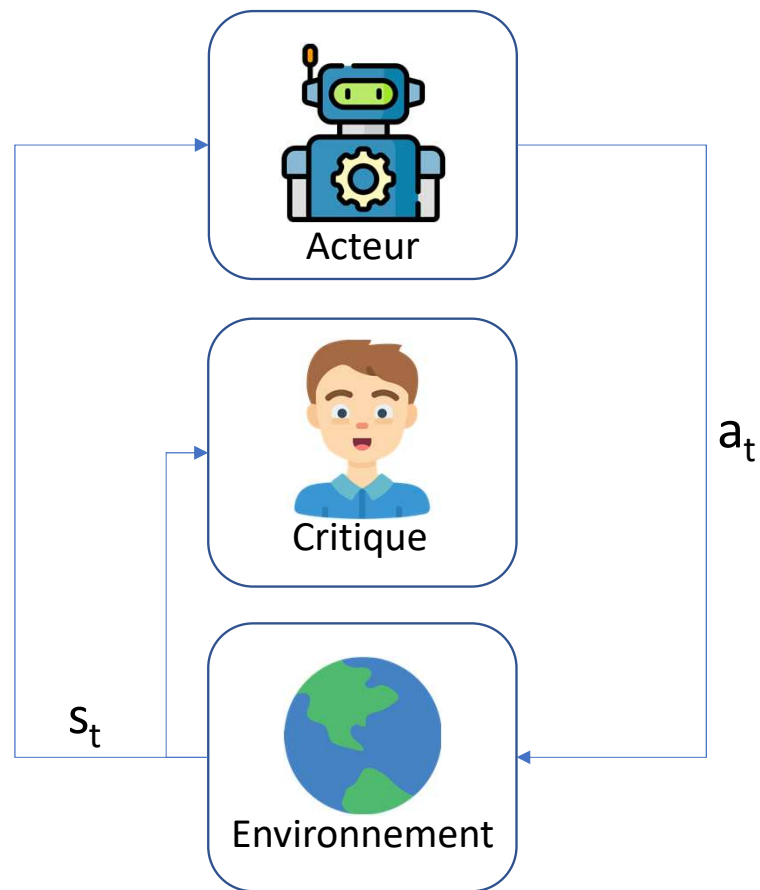
○ Principe:





Approche Acteur-Critique

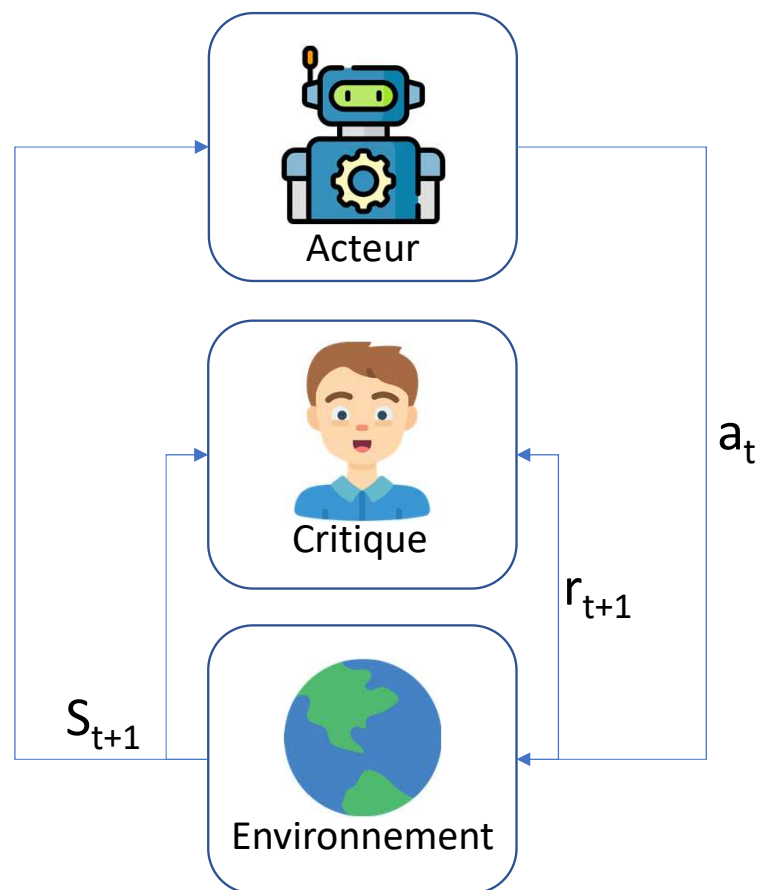
○ Principe:





Approche Acteur-Critique

○ Principe:

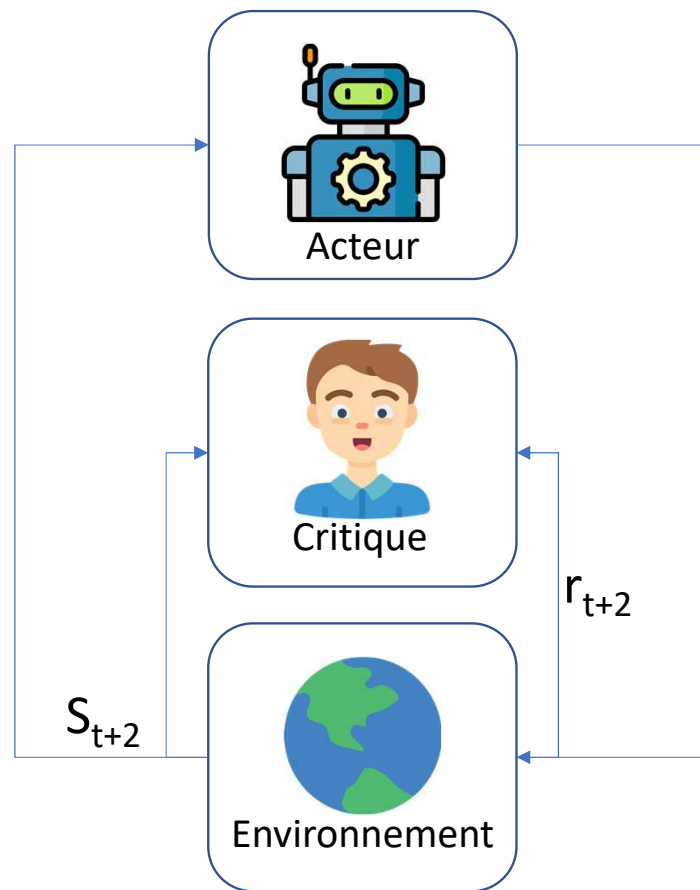


- Estimation de la fonction de valeur par le critique
- L'acteur met à jour ses poids



Approche Acteur-Critique

○ Principe:



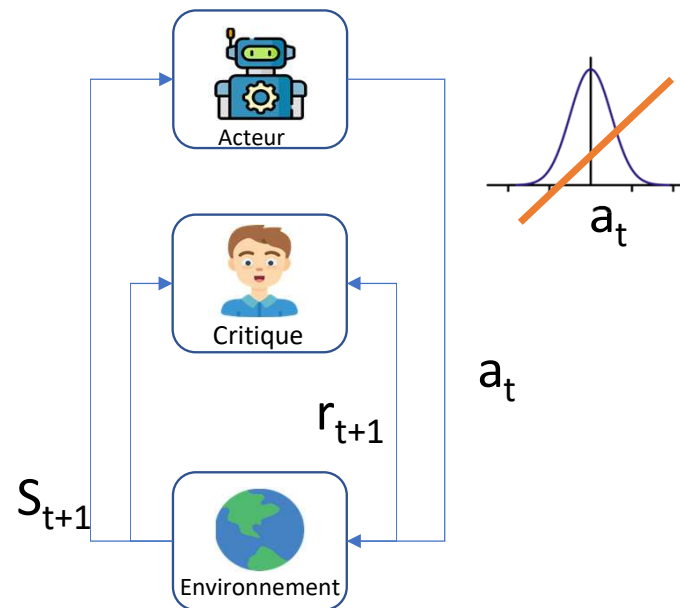
a_{t+1} - Le critique met à jour ses poids



Approche Deep Deterministic Policy Gradient (DDPG)

○ Principe:

- Approche acteur-critique où l'acteur génère une action au lieu d'une probabilité sur les actions
- Utilisation de réseaux cibles pour l'acteur et le critique
- Exploration via ajout d'un bruit sur l'action générée





Approche Deep Deterministic Policy Gradient (DDPG)

◉ Algorithme:

- Initialiser le réseau du critique $Q(s, a|\theta^Q)$ et de l'acteur $\mu(s|\theta^\mu)$
- Initialiser les réseaux cibles Q' et μ' avec les poids $\theta^{Q'} = \theta^Q$ et $\theta^{\mu'} = \theta^\mu$
- Initialiser un buffer R
- Pour $episode = 1, \dots, M$:
 - Collecter l'état s_0
 - Pour $t = 1, \dots, T$:
 - Choisir l'action $a_t = \mu(s|\theta^\mu) + \aleph_t$
 - Exécuter l'action a_t et collecter la récompense r_t et l'état s_{t+1}
 - Stocker la transition (s_t, a_t, r_t, s_{t+1}) dans R
 - Prendre aléatoirement un minibatch N transitions (s_i, a_i, r_i, s_{i+1}) R
 - $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
 - Mettre à jour le critique ne minimisant $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
 - Mettre à jour la politique de l'acteur en utilisant: $\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$
 - Mettre à jour les réseaux cibles:

$$\theta^{Q'} = \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

$$\theta^{\mu'} = \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$



Exemple avec l'environnement Gym

Package Gym:

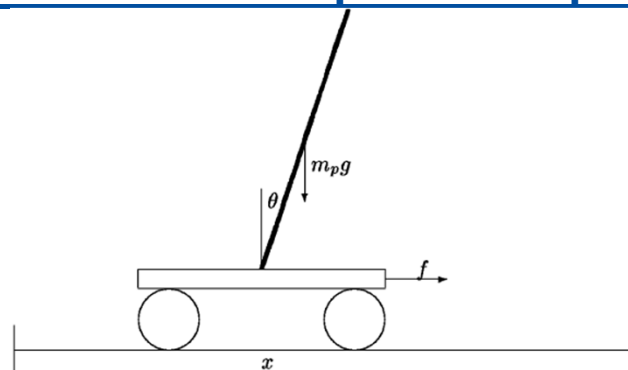
- Package développé par OpenAI (<https://gym.openai.com/>)
- Propose des environnements (jeux Atari, atterrissage sonde lunaire, car racing,...) permettant de tester les approches d'apprentissage par renforcement
- Possibilité de personnaliser son propre environnement via l'implémentation de méthodes spécifiques



Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

Test d'une approche acteur-critique sur un pendule inversé:

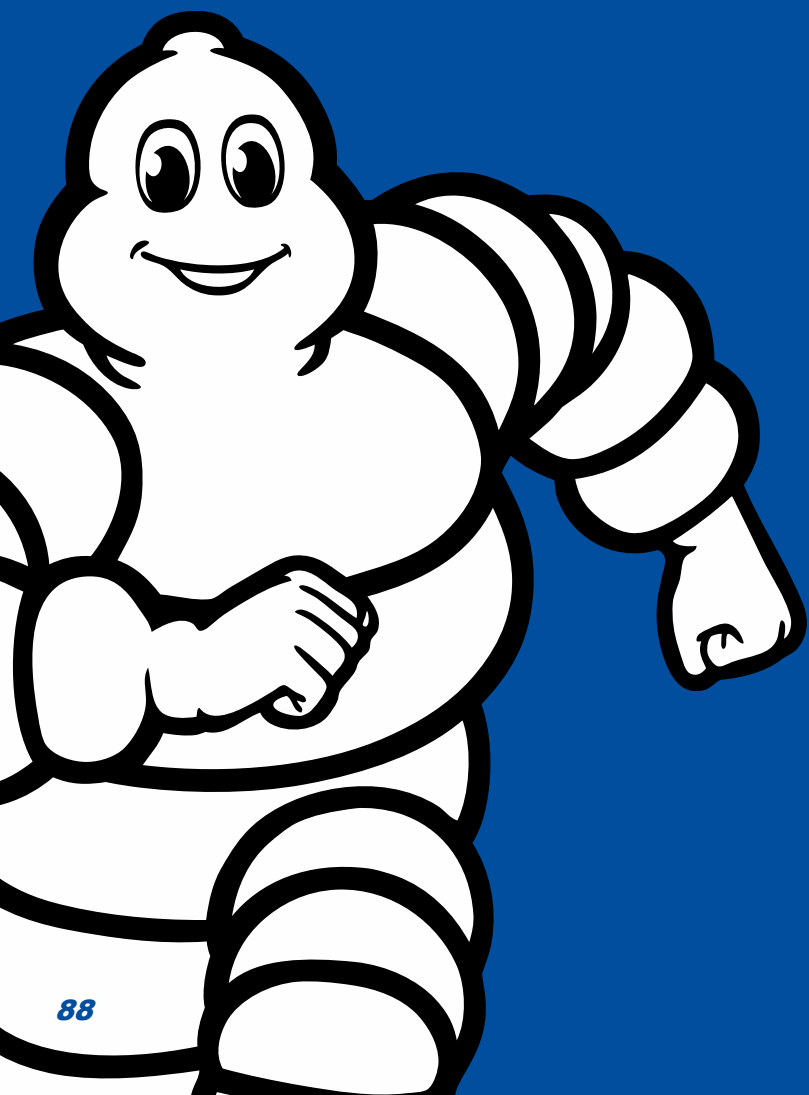




Conclusion ***Apprentissage par Renforcement***

***A vous de vous
exprimer...***





MOTION FOR LIFE

88

