

# Simulation abstraite : une analyse statique de modèles Simulink

Alexandre Chapoutot<sup>1</sup>

Laboratoire MeASI - CEA LIST

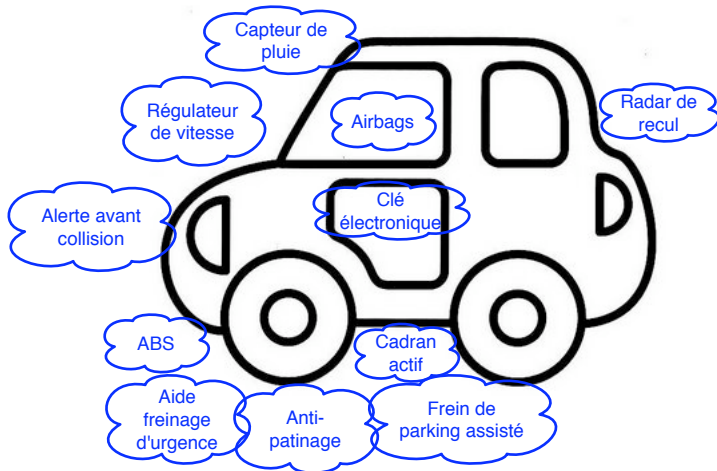
Soutenance de thèse  
Ecole Polytechnique, Palaiseau  
8 décembre 2008

---

<sup>1</sup>Sous la direction de Matthieu Martel

# Omniprésence des systèmes informatiques

Exemple : l'évolution de l'automobile

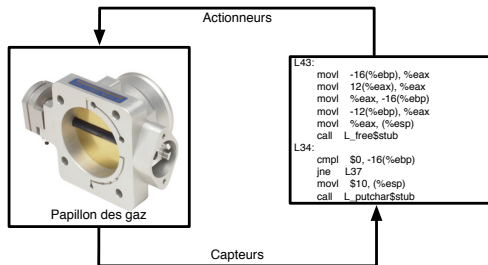




# Objets étudiés

## Etude des systèmes de contrôle-commande

Exemple : régulation du papillon des gaz



Quatre composants :

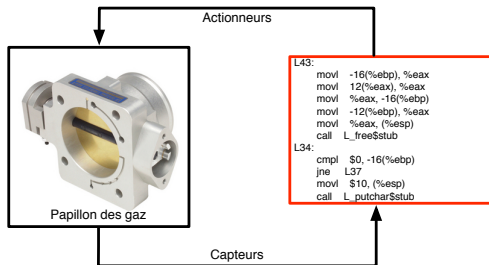
- Un programme (élément de contrôle)
- Un système physique (élément à contrôler)
- Des actionneurs (action sur le système physique)
- Des capteurs (mesure du système physique)

Hétérogénéité des composants : **systèmes hybrides**

# Objets étudiés

## Etude des systèmes de contrôle-commande

Exemple : régulation du papillon des gaz



Quatre composants :

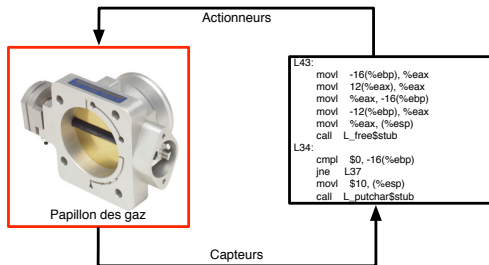
- **Un programme** (élément de contrôle)
- Un système physique (élément à contrôler)
- Des actionneurs (action sur le système physique)
- Des capteurs (mesure du système physique)

Hétérogénéité des composants : **systèmes hybrides**

# Objets étudiés

## Etude des systèmes de contrôle-commande

Exemple : régulation du papillon des gaz



Quatre composants :

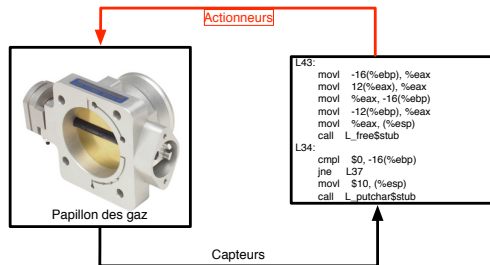
- Un programme (élément de contrôle)
- **Un système physique** (élément à contrôler)
- Des actionneurs (action sur le système physique)
- Des capteurs (mesure du système physique)

Hétérogénéité des composants : **systèmes hybrides**

# Objets étudiés

## Etude des systèmes de contrôle-commande

Exemple : régulation du papillon des gaz



Quatre composants :

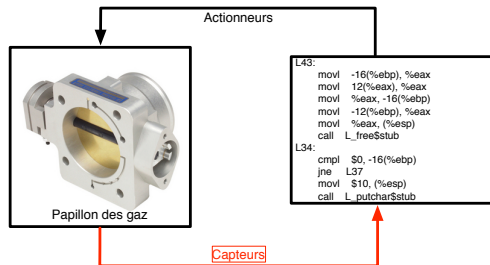
- Un programme (élément de contrôle)
- Un système physique (élément à contrôler)
- **Des actionneurs** (action sur le système physique)
- Des capteurs (mesure du système physique)

Hétérogénéité des composants : **systèmes hybrides**

# Objets étudiés

## Etude des systèmes de contrôle-commande

Exemple : régulation du papillon des gaz



Quatre composants :

- Un programme (élément de contrôle)
- Un système physique (élément à contrôler)
- Des actionneurs (action sur le système physique)
- **Des capteurs** (mesure du système physique)

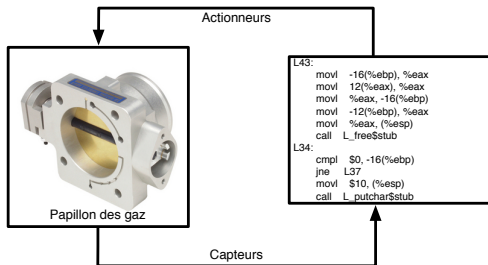
Hétérogénéité des composants : **systèmes hybrides**



# Objets étudiés

## Etude des systèmes de contrôle-commande

Exemple : régulation du papillon des gaz

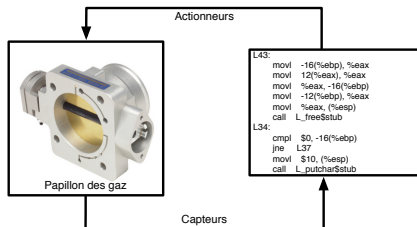


Quatre composants :

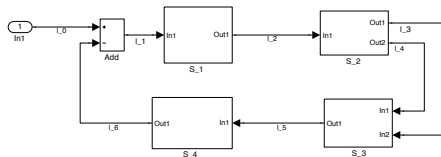
- Un programme (élément de contrôle)
- Un système physique (élément à contrôler)
- Des actionneurs (action sur le système physique)
- Des capteurs (mesure du système physique)

Hétérogénéité des composants : **systèmes hybrides**

# Modélisation en Simulink

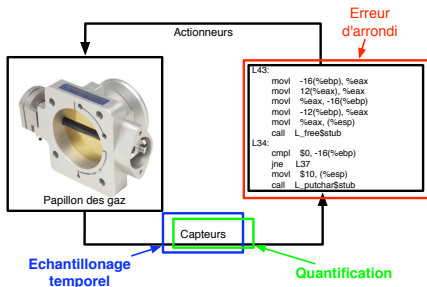


Modèle Simulink : programme décrivant le système hybride

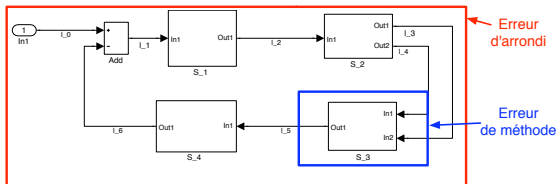


# Approximations numériques

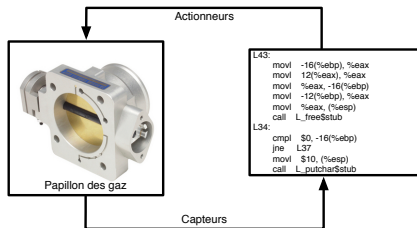
## Place des approximations dans le système réel



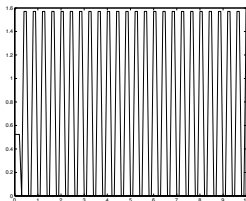
## Place des approximations dans le modèle Simulink



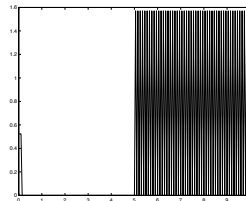
# Vérification formelle de la modélisation en Simulink



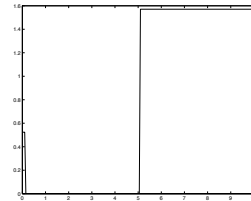
Simulation numérique du papillon des gaz (fermeture puis ouverture)



Euler

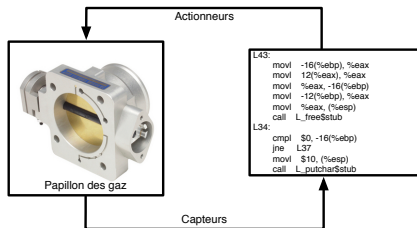


RK4

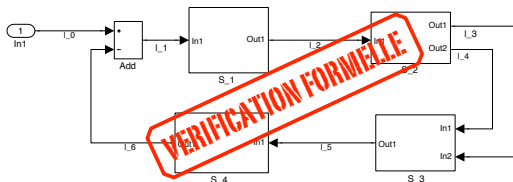


Newton + extrapolation

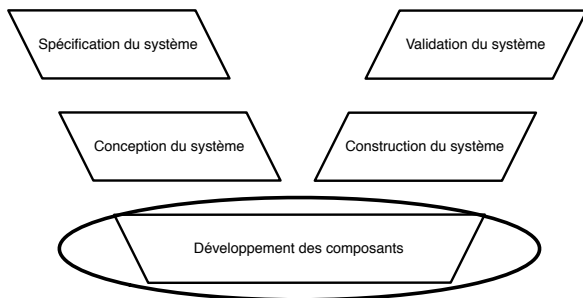
# Vérification formelle de la modélisation en Simulink



Modèle Simulink : programme décrivant le système hybride



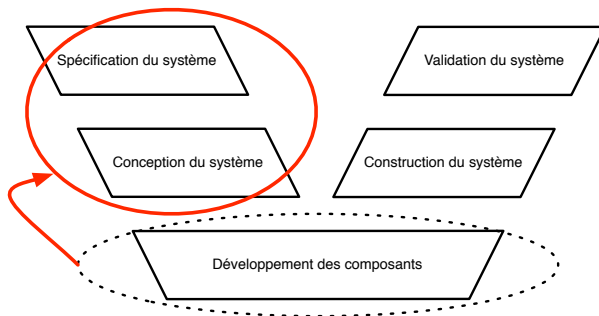
# Vérification formelle appliquée au niveau du code



## Analyse statique par interprétation abstraite

Analyseurs	Langages	Propriétés
Astrée (ENS)	C	Erreurs à l'exécution
Polyspace (Mathworks)	C, C++, Ada	Erreurs à l'exécution
aiT (AbsInt)	assembleur	Pire temps d'exécution
C Global Surveyor (NASA)	C	Pointeurs
Airac5 (Corée)	C	Débordements mémoire
Fluctuat (CEA)	C, assembleur	Précision numérique

# Objectif du travail de thèse



- Appliquer les méthodes de vérification formelle au plus tôt dans le cycle de développement
- Valider la méthode de résolution et pas sa mise en œuvre : simplification
- Valider en mimant au mieux les conditions réelles d'exécution

# Contributions : simulation abstraite

Analyse statique par interprétation abstraite de modèles Simulink  
Pour l'étude de la précision numérique

## Problèmes

- Approximations numériques importantes
- Sémantique de Simulink pas clairement définie

## Contributions théoriques

- Définition de domaines numériques abstraits [AFADL'07, TSI'07]
- Définition de la sémantique d'un sous-ensemble de Simulink [LCTES'06, SLA++P'08, soumis]

## Contribution pratique

Prototype d'analyseur statique de modèles Simulink [SLA++P'08]



# Plan de l'exposé

- 1 **Modèles Simulink**
  - Un outil de modélisation
  - Présentation du langage
  - Sémantique et simulation numérique
- 2 **Simulation abstraite**
  - Domaines numériques abstraits
  - Domaine des séquences
  - Analyse statique de Simulink
- 3 **Expérimentations**
  - Architecture logicielle du simulateur abstrait
  - Expérimentation : pédale de frein
  - Expérimentation : papillon des gaz
- 4 **Conclusion et perspectives**
  - Conclusion
  - Perspectives

# Plan de l'exposé

- 1 **Modèles Simulink**
  - Un outil de modélisation
  - Présentation du langage
  - Sémantique et simulation numérique
- 2 **Simulation abstraite**
  - Domaines numériques abstraits
  - Domaine des séquences
  - Analyse statique de Simulink
- 3 **Expérimentations**
  - Architecture logicielle du simulateur abstrait
  - Expérimentation : pédale de frein
  - Expérimentation : papillon des gaz
- 4 **Conclusion et perspectives**
  - Conclusion
  - Perspectives

# Simulink en bref

## Généralités

- Extension de Matlab
- Environnement graphique utilisé pour la conception de systèmes de contrôle-commande
- Plate-forme de simulation numérique
- Nombreuses bibliothèques dédiées : traitement du signal, systèmes physiques, automobile, etc.
- Standard de fait dans l'industrie (automobile en particulier)

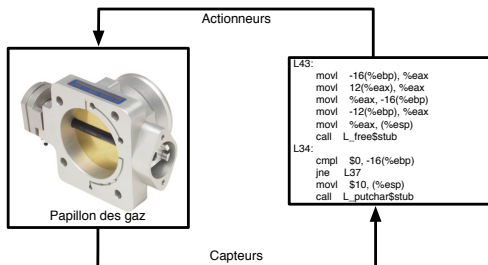
## Paramétrage et fonctionnalités

- Formats des données variés
- Algorithmes numériques : Euler, Runge-Kutta, etc.
- Typeur, débogueur, générateur de code

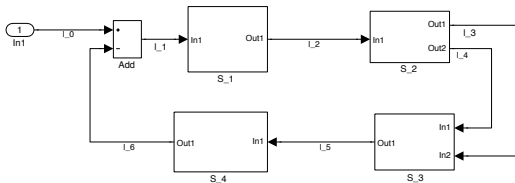
## Diversité des modèles

- A temps continu
- A temps discret
- Hybrides

# Modélisation Simulink

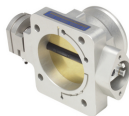
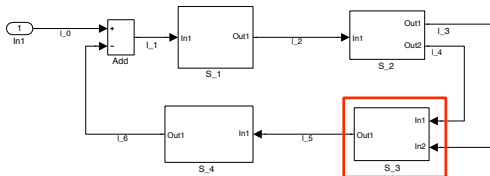


## Modèle Simulink



# Modèle du papillon des gaz

## Modèle Simulink complet

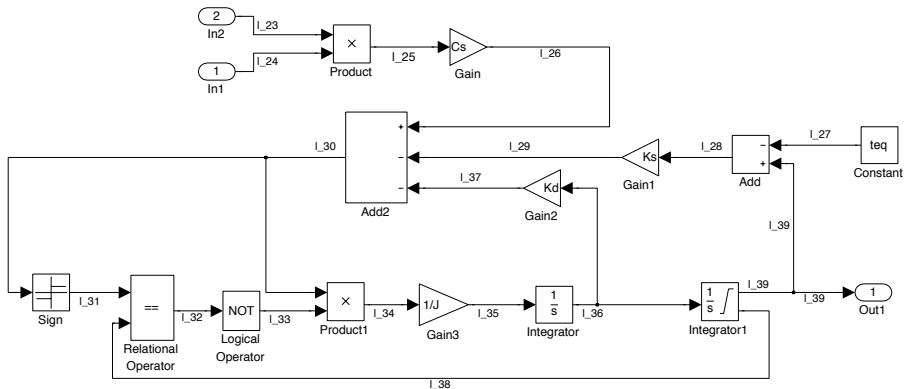


## Modèle mathématique

$$\left\{ \begin{array}{l} T(t) = \text{Direction} \times \text{Effort} \times C_s \\ \dot{\omega}(t) = \frac{1}{J} (-K_s(\theta(t) - \theta_{eq}) - K_d\omega(t) + T(t)) \quad 0 < \theta < \pi/2 \\ \text{si } (\theta < 0 \wedge \text{sgn}(\dot{\omega}(t) = -1)) \vee ((\theta > \pi/2 \wedge \text{sgn}(\dot{\omega}(t) = 1))) \\ \text{alors } \dot{\omega}(t) = 0 \end{array} \right.$$

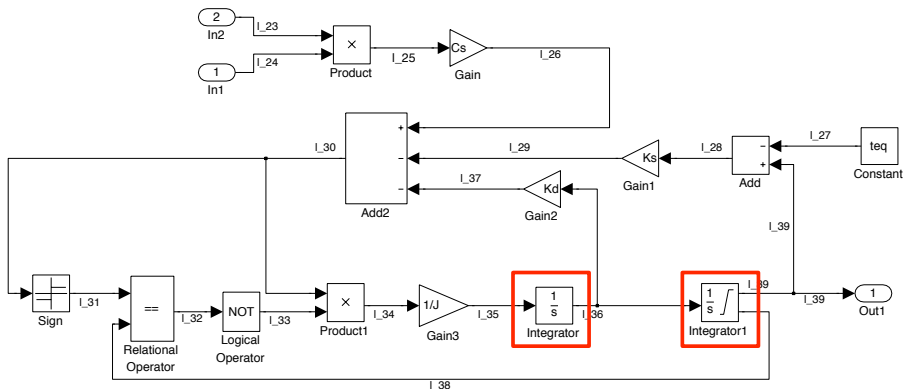
# Modèle du papillon des gaz

## Modèle Simulink (système à temps continu)



# Modèle du papillon des gaz

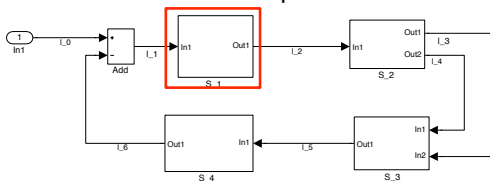
## Modèle Simulink (système à temps continu)



## Bloc temporel : Integrator (temps continu)

# Modèle du régulateur

## Modèle Simulink complet



```

L43:
movl  -16(%ebp), %eax
movl  12(%eax), %eax
movl  %eax, -16(%ebp)
movl  -12(%ebp), %eax
movl  %eax, (%esp)
call  L_free$stub

L34:
cmpl  $0, -16(%ebp)
jne   L37
movl  $10, (%esp)
call  L_putchar$stub
  
```

## Modèle mathématique

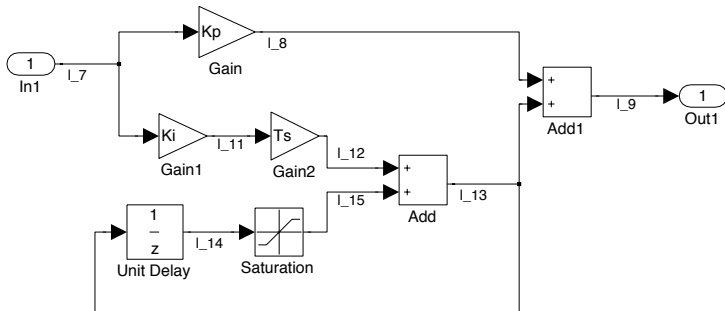
$$\begin{cases} y(k) = y_p(k) + y_i(k) \\ y_p(k) = K_p e(k) \\ y_i(k+1) = y_i(k) + K_i T_s e(k) \quad 0 < y_i(k) < 1 \end{cases}$$

## Régulateur PI (Proportionnel-Intégral)



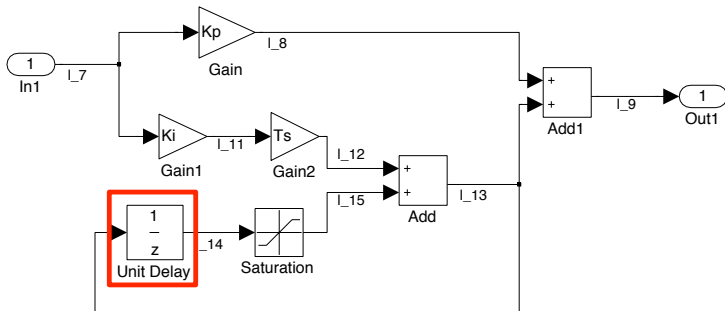
# Modèle du régulateur

Modèle du contrôleur (système à temps discret)



# Modèle du régulateur

Modèle du contrôleur (système à temps discret)

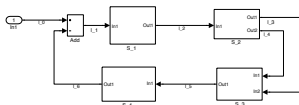


Bloc temporel : Unit Delay (temps discret)

# Simulation numérique : sémantique de Simulink

## Simulation numérique

### Modèle Simulink



1: Etats initiaux

2: **répéter**

3: Lire les entrées

4: Calculer les sorties

5: Calculer les états

6: Calculer le prochain pas de temps

7: **jusqu'à** Temps fin de simulation

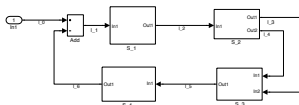
Etat : "valeurs des précédentes itérations nécessaires pour calculer la sortie courante" (continu vs discret)

Description insuffisante pour la vérification formelle

# Simulation numérique : sémantique de Simulink

## Simulation numérique

### Modèle Simulink



1: Etats initiaux

2: **répéter**

3: Lire les entrées

4: Calculer les sorties

5: Calculer les états






6: Calculer le prochain pas de temps

7: **jusqu'à** Temps fin de simulation

Etat : "valeurs des précédentes itérations nécessaires pour calculer la sortie courante" (continu vs discret)

Description insuffisante pour la vérification formelle

# Formalisation






Nom	Bloc	Equations
Constant	 Constant	$\{l_1 = c, \emptyset\}$
Add	 Add	$\{l_3 = l_1 + l_2, \emptyset\}$
Switch	 Switch	$\{l_4 = \text{if}(p(l_1), l_2, l_3), \emptyset\}$
Integrator	 Integrator	$\{l_2(t) = \sigma(t), \dot{\sigma}(t) = l_1(t)\}$
UnitDelay	 Unit Delay	$\{l_2(k) = \sigma(k), \sigma(k+1) = l_1(k)\}$

Deux types d'équations :

- équations liées aux sorties
- équations liées aux états : opérateur temporel dans la sémantique

Note : Utilisation de l'équivalence  $y(t) = \int x(z)dz \equiv \dot{y}(t) = x(t)$

# Formalisation






Nom	Bloc	Equations
Constant	 Constant	$\{l_1 = c, \emptyset\}$
Add	 Add	$\{l_3 = l_1 + l_2, \emptyset\}$
Switch	 Switch	$\{l_4 = \text{if}(p(l_1), l_2, l_3), \emptyset\}$
Integrator	 Integrator	$\{l_2(t) = \sigma(t), \dot{\sigma}(t) = l_1(t)\}$
UnitDelay	 Unit Delay	$\{l_2(k) = \sigma(k), \sigma(k+1) = l_1(k)\}$

Deux types d'équations :

- équations liées aux sorties
- équations liées aux états : opérateur temporel dans la sémantique

Note : Utilisation de l'équivalence  $y(t) = \int x(z)dz \equiv \dot{y}(t) = x(t)$

# Formalisation

Nom	Bloc	Equations
Constant	 Constant	$\{l_1 = c, \emptyset\}$
Add	 Add	$\{l_3 = l_1 + l_2, \emptyset\}$
Switch	 Switch	$\{l_4 = \text{if}(p(l_1), l_2, l_3), \emptyset\}$
Integrator	 Integrator	$\{l_2(t) = \sigma(t), \dot{\sigma}(t) = l_1(t)\}$
UnitDelay	 Unit Delay	$\{l_2(k) = \sigma(k), \sigma(k+1) = l_1(k)\}$

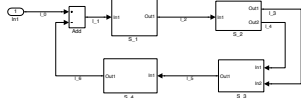
Deux types d'équations :

- équations liées aux sorties
- **équations liées aux états** : opérateur temporel dans la sémantique

Note : Utilisation de l'équivalence  $y(t) = \int x(z)dz \equiv \dot{y}(t) = x(t)$

# Formalisation : sémantique de Simulink

## Modèle Simulink



## Système d'équations

$$\left\{ \begin{array}{l} l_0 = \text{In1} \\ l_1 = l_0 - l_6 \\ l_2 = S_1(l_1) \\ (l_3, l_4) = S_2(l_2) \\ l_5 = S_3(l_3, l_4) \\ l_6 = S_4(l_5) \end{array} \right\}$$

## Simulation numérique

- 1: Etats initiaux
- 2: **répéter**
- 3: Lire les entrées
- 4: Calculer les sorties
- 5: Calculer les états
- 6: Calculer le prochain pas de temps
- 7: **jusqu'à** Temps fin de simulation

Discrétise les fonctions continues

$$\dot{y}(t) = x(t)$$

transformé en

$$\eta(k+1) = \text{solver}(\eta(k), x(k))$$

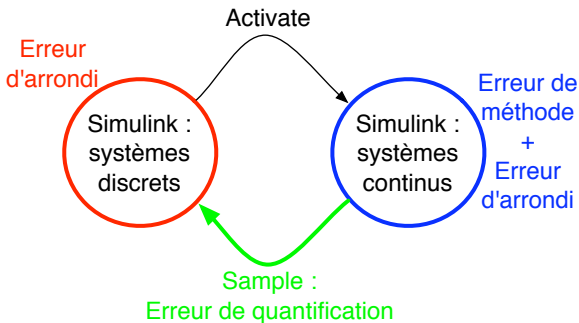
$$y(k) = \eta(k)$$

Constat : plusieurs méthodes d'intégration donc plusieurs sémantiques



# Formalisation : propriétés numériques

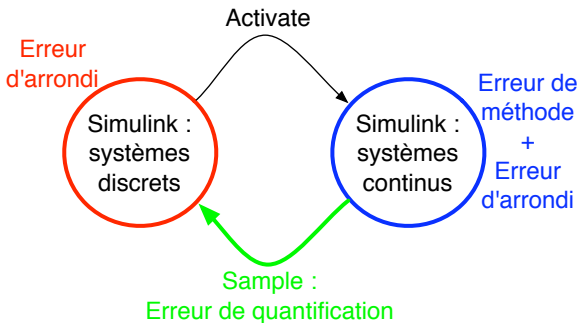
Objectif : calculer la distance entre les comportements mathématiques et la simulation numérique (critère de correction)



Résultat : Evaluation de la robustesse des programmes aux approximations numériques

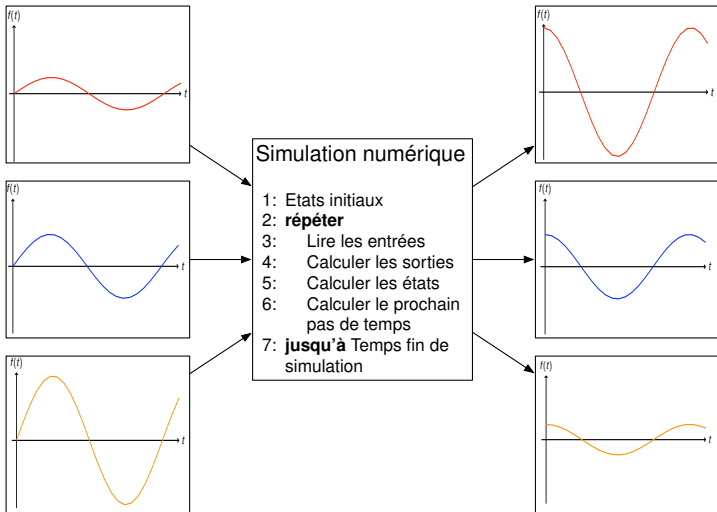
# Formalisation : propriétés numériques

Objectif : calculer la distance entre les comportements mathématiques et la simulation numérique (critère de correction)

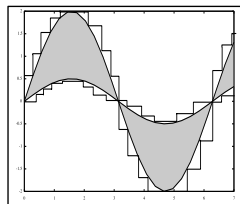


Résultat : Evaluation de la robustesse des programmes aux approximations numériques **pour un ensemble d'entrées**

# Simulation abstraite

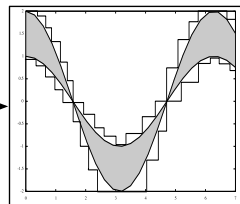


# Simulation abstraite



## Simulation abstraite

- 1: Etats initiaux
- 2: **tant que** Pas point fixe **faire**
- 3: Lire les entrées
- 4: Calculer les sorties
- 5: Calculer les états
- 6: Calculer le prochain pas de temps
- 7: **fin tant que**



Approche sémantique des modèles Simulink.

**Modélisation sémantique** : sémantique des séquences [Kahn'74]

# Plan de l'exposé

- 1 **Modèles Simulink**
  - Un outil de modélisation
  - Présentation du langage
  - Sémantique et simulation numérique
- 2 **Simulation abstraite**
  - Domaines numériques abstraits
  - Domaine des séquences
  - Analyse statique de Simulink
- 3 **Expérimentations**
  - Architecture logicielle du simulateur abstrait
  - Expérimentation : pédale de frein
  - Expérimentation : papillon des gaz
- 4 **Conclusion et perspectives**
  - Conclusion
  - Perspectives

# Manipulation d'ensembles de valeurs

## Intervalles [Moore'66] [Cousot&Cousot'77]

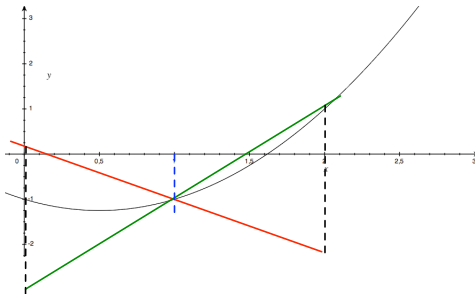
$$[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$$

Problème de dépendance :  $X - X \neq 0$  en général

## Solution : forme centrée

Application du théorème des accroissements finis

$$\mathbf{f}([a, b]) \subseteq \mathbf{f}(m) + [\mathbf{f}']([a, b])([a, b] - m) \quad m \text{ milieu de l'intervalle } [a, b]$$



# Manipulation d'ensembles de valeurs

## Intervalles [Moore'66] [Cousot&Cousot'77]

$$[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$$

Problème de dépendance :  $X - X \neq 0$  en général

## Solution : forme centrée

Application du théorème des accroissements finis

$$\mathbf{f}([a, b]) \subseteq \mathbf{f}(m) + [\mathbf{f}']([a, b])([a, b] - m)$$

## Généralisation : forme de Taylor

$$\mathbf{f}([a, b]) \subseteq \mathbf{f}(m) + \mathbf{f}'(m)([a, b] - m) + \dots + \mathbf{f}^{n-1}(m) \frac{([a, b] - m)^{n-1}}{(n-1)!} + [\mathbf{f}^n]_n([a, b]) \frac{([a, b] - m)^n}{n!}$$

$m$  milieu de l'intervalle  $[a, b]$

# Contribution : domaine numérique abstrait

## Domaine numérique abstrait

L'ensemble des formes de Taylor forme un domaine numérique abstrait  $\langle \mathcal{T}, \sqsubseteq_{\mathcal{T}}, \perp, \top, \sqcup, \sqcap \rangle$

Intérêt : prise en compte des relations entre les variables (valeurs des dérivées partielles)

Deux applications :

- Adaptation des algorithmes d'intégration numérique (par exemple Euler, RK) avec du calcul par intervalles
- Amélioration de l'évaluation des erreurs d'arrondi



# Amélioration du calcul des erreurs d'arrondi

## Domaine des flottants avec erreurs [Goubault'01, Martel'02]

Défini spécifiquement pour estimer les erreurs de calculs

## Principe : décomposition d'une valeur réelle

- Valeur flottante
- Erreur d'arrondi : distance entre réel et flottant ( $\downarrow$ )

## Règles de calcul des erreurs

$$a = (f_a, e_a) \text{ et } b = (f_b, e_b)$$

$$a + b = (f_a +_F f_b, e_a + e_b + \downarrow (f_a + f_b))$$

$$a \times b = (f_a \times_F f_b, e_a f_b + e_b f_a + e_a e_b + \downarrow (f_a \times f_b))$$

# Amélioration du calcul des erreurs d'arrondi

## Domaine des flottants avec erreurs différenciées

Combinaison du domaine des flottants avec erreurs et des formes de Taylor [AFADL'07, TSI'07]

Abstraction des erreurs fondée sur le domaine des formes de Taylor  
Permet de prendre en compte les dépendances entre les erreurs

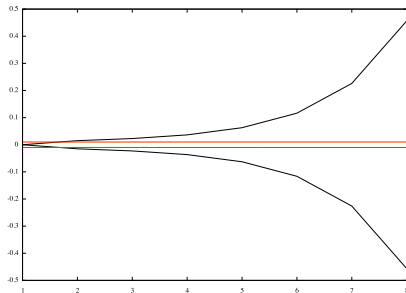
Exemple : racine carrée par méthode de Newton

```

double xn = 0.1;
double xn1 = 0.0;
double a = [25,25]_[-0.1,0.1];
int cond = 0;
double temp = 0.0;
double res = 0.0;

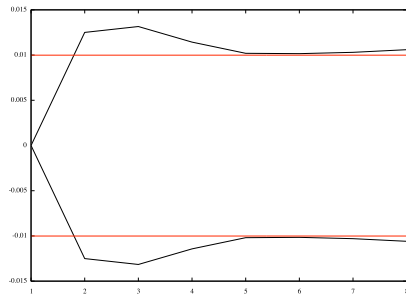
    while (cond < 1) {
        xn1 = 0.5 * xn * (3.0 - a * xn * xn);
        temp = xn1 - xn;
        if (temp < 1e-12) { cond = 1; }
        if (temp > -1e-12) { cond = 1; }
        xn = xn1;
        res = xn1 * a;
    }
  
```

# Evolution des erreurs



Abstraction par des intervalles :

- Intervalle d'erreurs croissant
- Conclusion : instabilité numérique



Abstraction par forme de Taylor :

- Intervalle d'erreurs stable
- Conclusion : stabilité numérique

# Plan de l'exposé

- 1 Modèles Simulink**
  - Un outil de modélisation
  - Présentation du langage
  - Sémantique et simulation numérique
- 2 Simulation abstraite**
  - Domaines numériques abstraits
  - **Domaine des séquences**
  - Analyse statique de Simulink
- 3 Expérimentations**
  - Architecture logicielle du simulateur abstrait
  - Expérimentation : pédale de frein
  - Expérimentation : papillon des gaz
- 4 Conclusion et perspectives**
  - Conclusion
  - Perspectives

# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$

	1	2	3	4	5	6	7	8	9	10	11	12	...
Séquence concrète	0	1	2	3	4	5	6	6	6	6	6	6	...

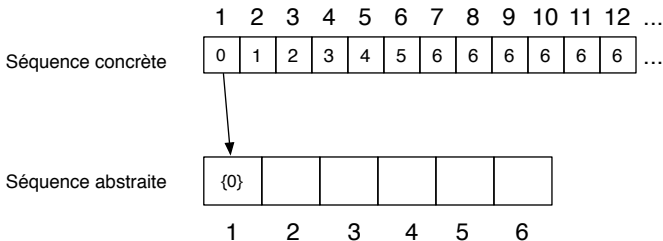
Séquence abstraite						
	1	2	3	4	5	6

# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$

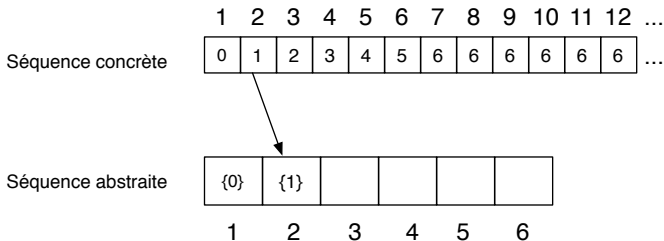


# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$

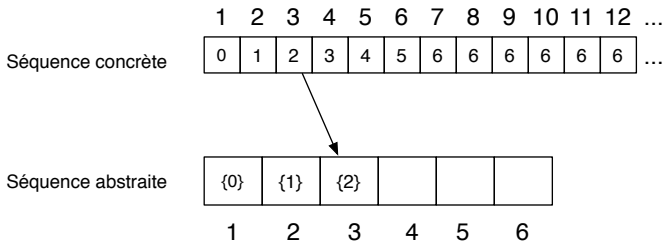


# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$



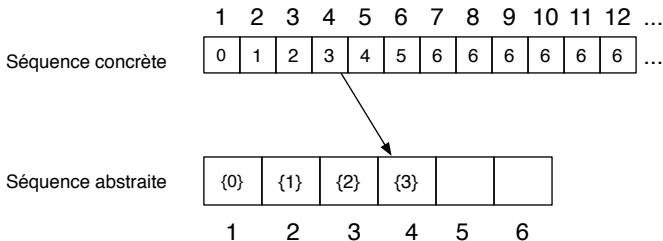


# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$

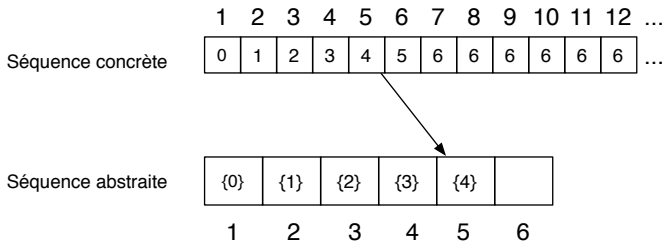


# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$

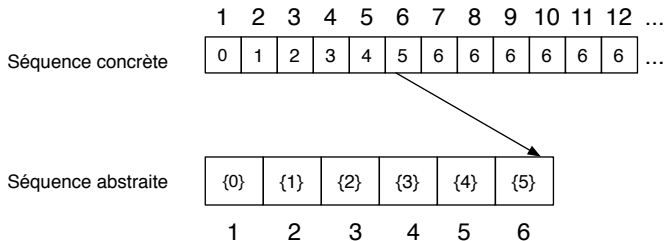


# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$

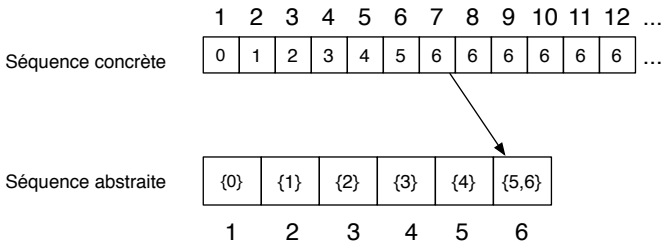


# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$

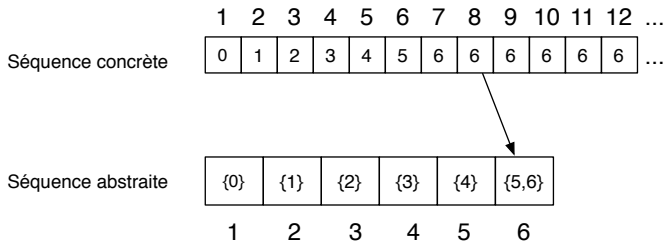


# Représentation des séquences

Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

Exemple de séquence abstraite avec la fonction de partition :

$$\mu(k) = \begin{cases} k & \text{si } k < 6 \\ 6 & \text{sinon} \end{cases}$$



# Représentation des séquences

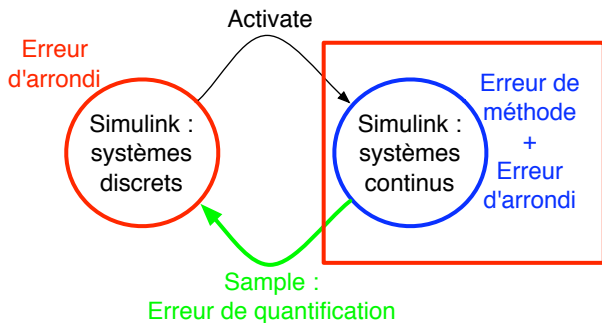
Algorithme de simulation construit une séquence de valeurs  
Représentation de séquences infinies de manière finie

## Domaine abstrait

Le domaine des séquences est un domaine abstrait paramétré par une fonction de partition.

- Prise en compte des propriétés des fonctions : périodicité
- Définition d'une notion de précision

# Mesure des approximations de la partie continue



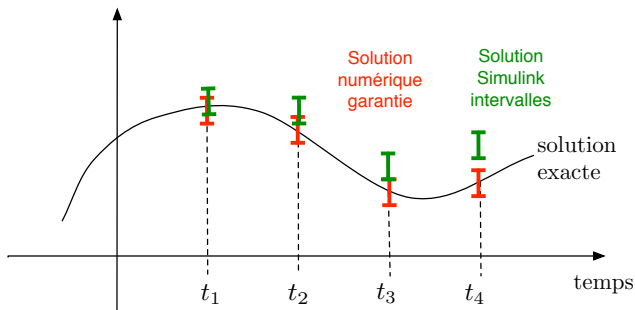
Evaluer la distance entre :

- la simulation numérique
- et un capteur parfait (discrétisation idéale du modèle mathématique)

# Mesure des approximations de la partie continue

Pour un système d'équations différentielles :

- Simulink : séquence d'intervalles approchant la solution
- Solution garantie : séquence d'intervalles encadrant la solution

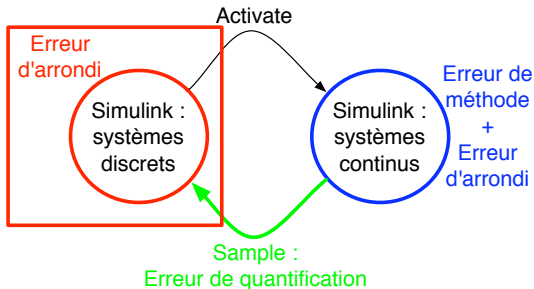


Critère de correction :

- distance entre la **solution garantie** et la **solution Simulink intervalle**



# Mesure des approximations de la partie discrète



Evaluer la distance entre :

- un calcul avec des nombres flottants
- et un calcul dans les réels

# Mesure des approximations de la partie discrète

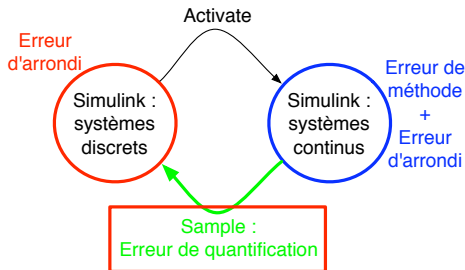
Donné directement par le domaine des flottants avec erreurs différenciées



Critère de correction :

- Plus la valeur de l'erreur est petite alors plus le résultat flottant est précis (c'est-à-dire proche du réel)

# Changement de sémantique



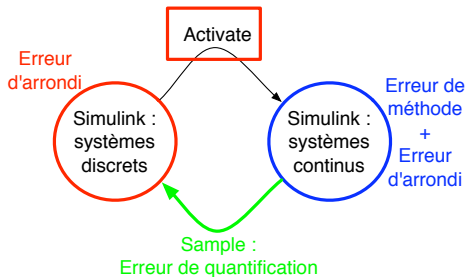
## Du continu vers le discret : sample

Transforme les valeurs de l'intégration numérique  $s$  et de l'intégration numérique garantie  $g$  en flottant avec erreurs

$$(s, g) \Rightarrow (s, g - s + q)$$

Introduit une erreur de quantification  $q$

# Changement de sémantique



## Du discret vers le continu : activate

Transforme une valeur flottante  $f$  avec erreur  $e$  en valeur simulation et valeur réelle

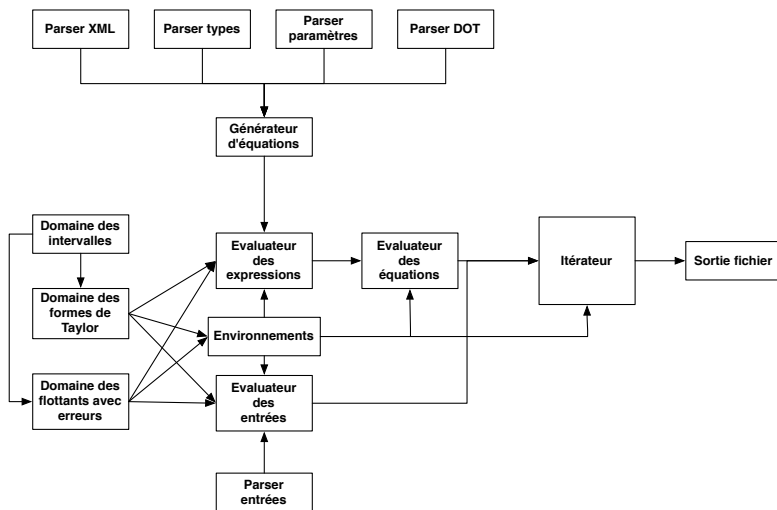
$$(f, e) \Rightarrow (f, f + e)$$

N'introduit pas d'approximation numérique supplémentaire

# Plan de l'exposé

- 1 Modèles Simulink
  - Un outil de modélisation
  - Présentation du langage
  - Sémantique et simulation numérique
- 2 Simulation abstraite
  - Domaines numériques abstraits
  - Domaine des séquences
  - Analyse statique de Simulink
- 3 Expérimentations
  - Architecture logicielle du simulateur abstrait
  - Expérimentation : pédale de frein
  - Expérimentation : papillon des gaz
- 4 Conclusion et perspectives
  - Conclusion
  - Perspectives

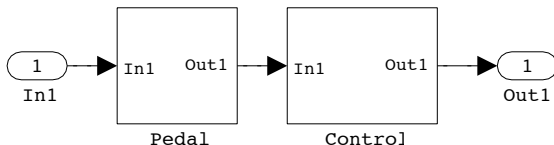
# Architecture logicielle



$O(6000)$  lignes OCaml,  $O(1000)$  lignes Matlab [Lim&Chapoutot'07]

# Expérimentation : pédale de frein

Système en boucle ouverte



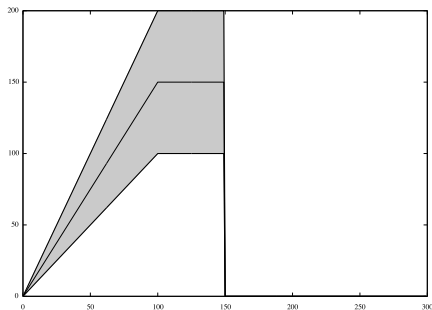
- Pédale : système masse-ressort-amortisseur
- Contrôle : détecteur de pression

Protocole expérimental :

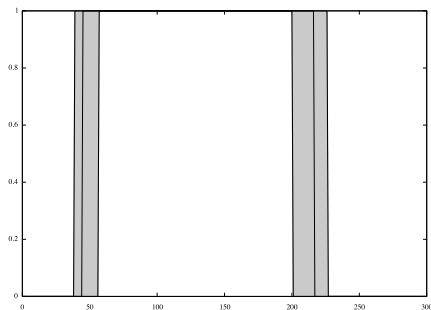
- Méthode d'intégration : Euler
- Pas d'intégration : 0.01

# Expérimentation : pédale de frein

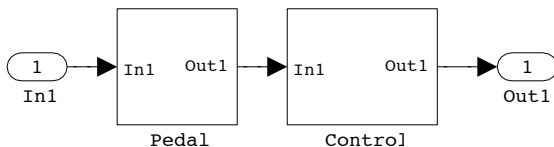
## Entrée du modèle



## Sortie du modèle



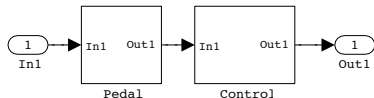
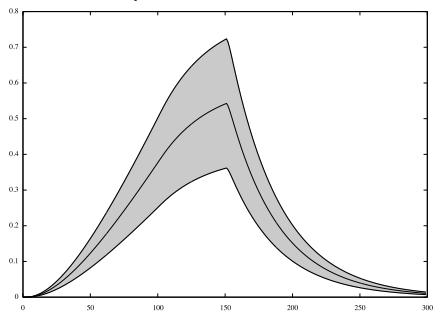
La pression de la pédale est détectée



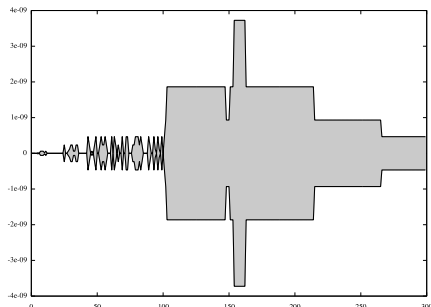


# Expérimentation : pédale de frein

## Sortie de la pédale

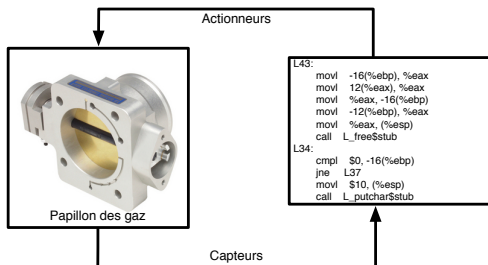


## Erreur de méthode et d'arrondi pour la pédale

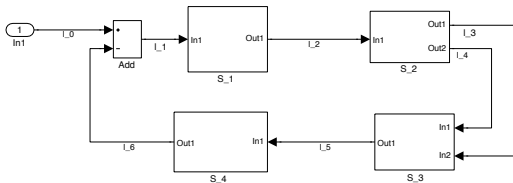


Conclusion : Les résultats de la simulation numérique sont très proches du modèle mathématique

# Expérimentation : papillon des gaz

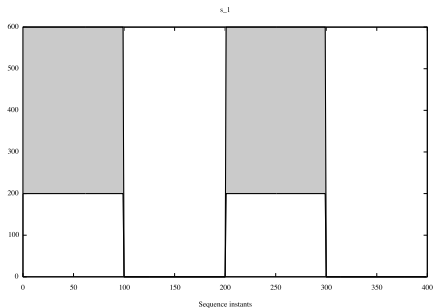


## Modèle Simulink

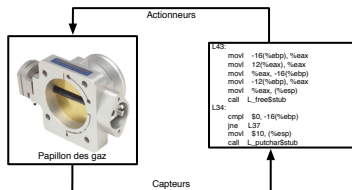
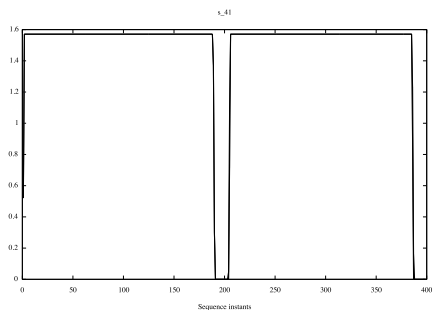


# Expérimentation : papillon des gaz

## Entrée du modèle

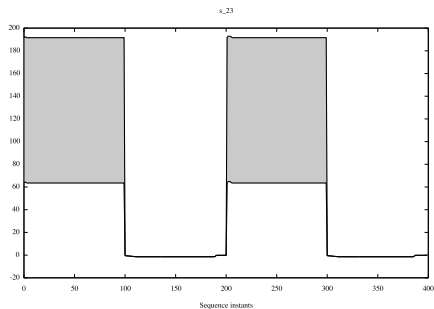


## Dynamique du papillon des gaz

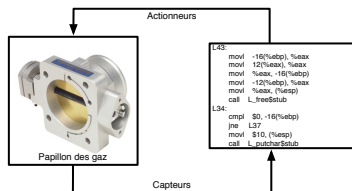
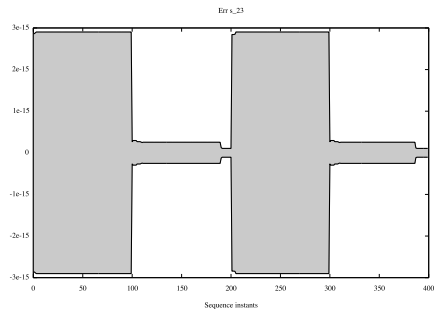


# Expérimentation : papillon des gaz

## Sortie du régulateur



## Erreurs numériques



Conclusion : Les résultats de la simulation numérique sont très proches du modèle mathématique

# Plan de l'exposé

## 1 Modèles Simulink

- Un outil de modélisation
- Présentation du langage
- Sémantique et simulation numérique

## 2 Simulation abstraite

- Domaines numériques abstraits
- Domaine des séquences
- Analyse statique de Simulink

## 3 Expérimentations

- Architecture logicielle du simulateur abstrait
- Expérimentation : pédale de frein
- Expérimentation : papillon des gaz

## 4 Conclusion et perspectives

- Conclusion
- Perspectives

# Conclusion

## Simulation abstraite

Méthode de validation automatique des spécifications de programmes de contrôle-commande avec :

- prise en compte de l'environnement physique
- prise en compte des approximations numériques

La simulation abstraite permet de séparer les problèmes :

- liés à la modélisation
- liés aux approximations numériques

# Perspectives

## Propriété temporelle

Quelles influences ont les approximations numériques sur le contrôle ?

## Outils spécifiques

Utilisation des outils issus de l'automatique [LCTES'06]  
En particulier, l'analyse fréquentielle

## Validation fonctionnelle multi-niveaux

Propagation de l'information des propriétés des spécifications  
au niveau du code

## Développement logiciel

Poursuivre le développement du simulateur abstrait