Improving Time Parallel Simulation for Monotone Systems

J.M. Fourneau

Laboratoire PRiSM, CNRS UMR 8144

Université de Versailles St-Quentin

joint work with N. Pekergin (Université Paris XII)

and I. Kadi and T.H. Dao Thi (PRiSM, UVSQ)

Distribuer des tranches de temps

- Chaque processeur simule une tranche de temps distincte
- Comment faire une trajectoire en regroupant les tranches ?
- Partie 1: Une approche générale : les calculs spéculatifs et des corrections éventuelles
- Partie 2 Des approches limitées mais plus rapides : propriétés stochastiques (régénération) permettant de deviner un état sans le simuler
- Partie 3: l'apport de la monotonie

Modèle de Simulation

- Boite noire déterministe réagissant à une séquence d'entrée.
- L'aléatoire est toujours pris en compte via la séquence d'entrée.

Partie 1: Calculs Spéculatifs

- Phase 0a : On divise le temps de simulation en K intervalles temporels égaux et on affecte chaque tranche à un processeur.
- Phase 0b : On choisit au hasard le point de démarrage de la simulation pour chaque tranche de temps.
- Phase 1 : on simule sur chaque processeur la tranche de temps indiquée. On conserve les trajectoires des générateurs uniformes utilisés.
- Il y a peu de chances que le point de départ de la simulation i (spéculatif) soit égal au point terminal de la simulation $i-1$.
- Mais cela peut arriver...

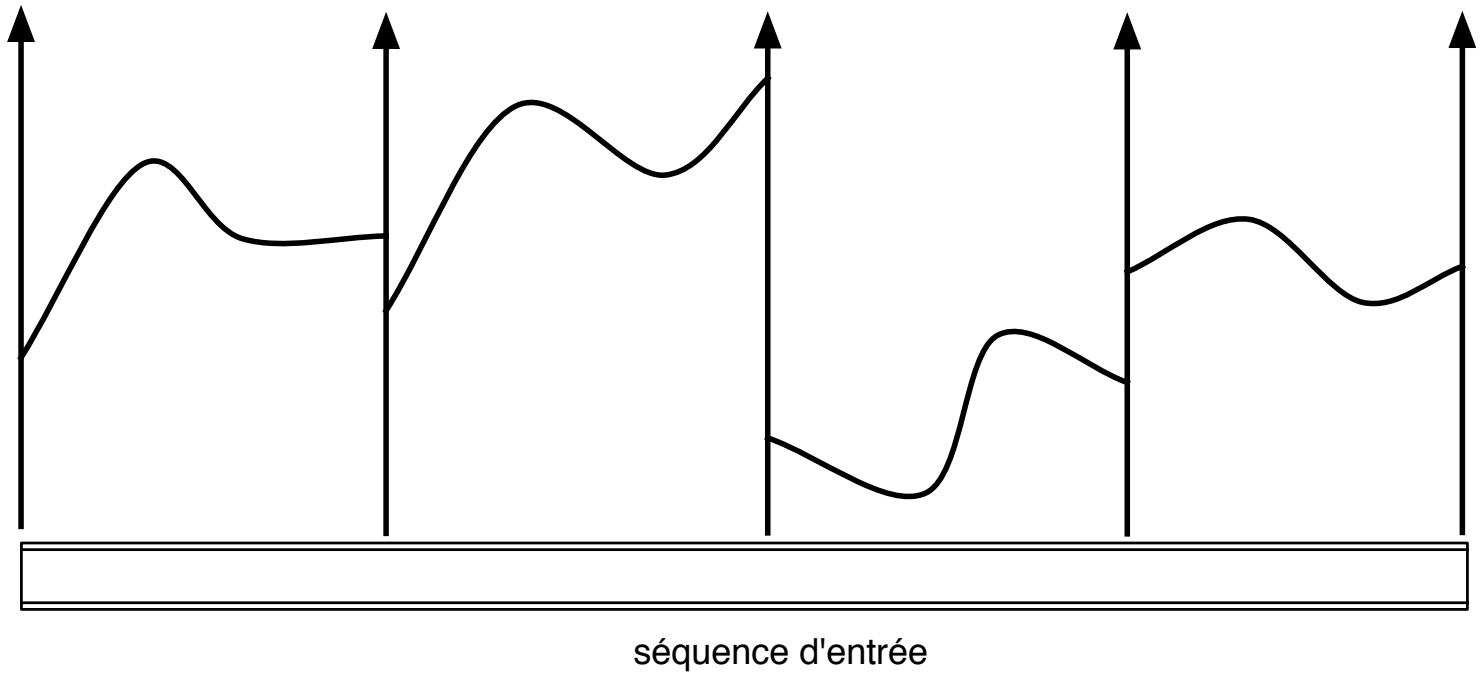

Figure 1: Simulation Temps parallèle sur 4 processeurs , phase 1

Corrections

- Phase 2 : Les processeurs 2 à K qui n'ont pas démarré correctement recommencent la simulation en utilisant le point de démarrage fourni par le processeur simulant la tranche de temps inférieure et en reemployant la trajectoire du générateur déjà construite.
- Si la simulation nouvelle couple avec la simulation ancienne, le calcul stoppe. Inutile de poursuivre, les deux simulations sont au même point en même temps et utiliseront les mêmes tirages de générateurs. La suite est donc la même...
- Si il n'y a pas couplage, le processus calcule un nouveau point de fin.

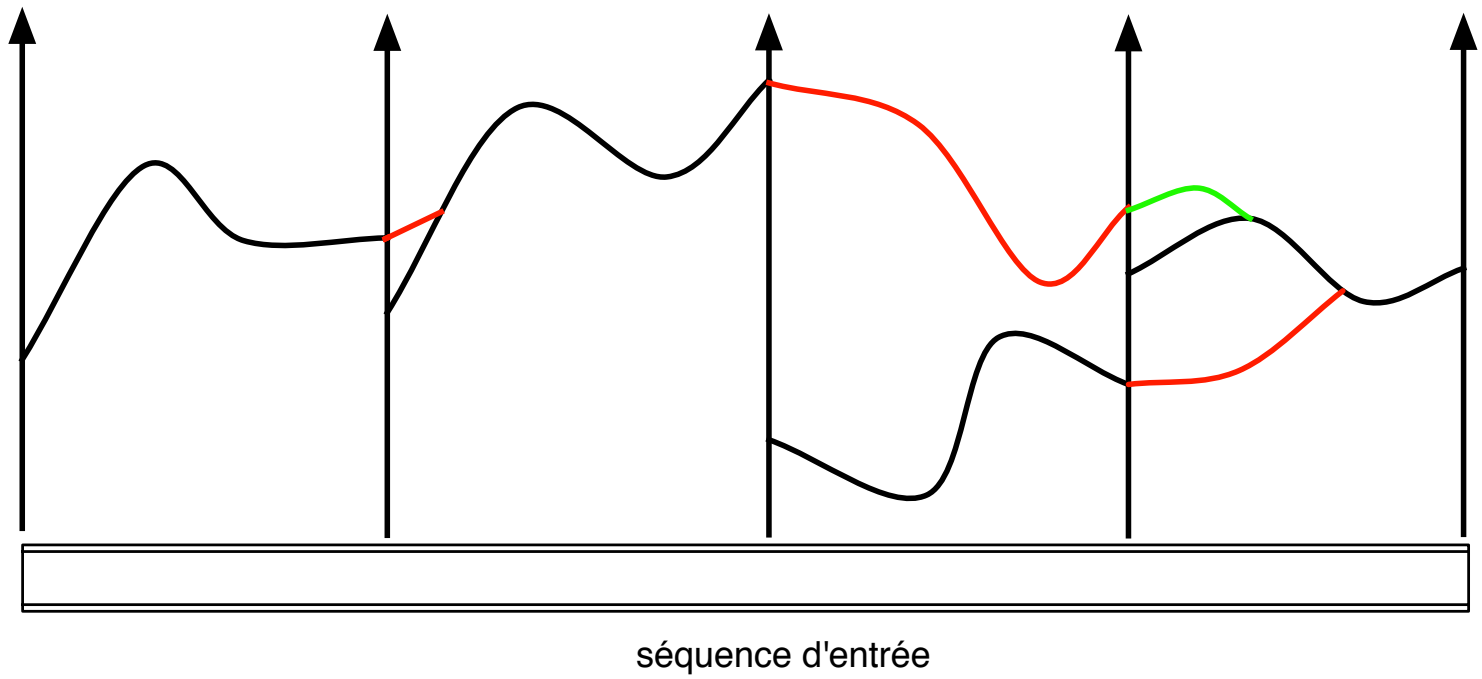

Figure 2: Simulation Temps parallèle sur 4 processeurs : phase 2 (rouge) et 3 (vert)

- Un processus a terminé si le processus qui simule la tranche précédent a terminé et si il a couplé.
- On recommence tant que tous les processus n'ont pas terminé.
- Après l'étape 1, le processus 1 a terminé. A la fin de l'étape 2, le processus 2 a sûrement terminé mais pour les autres, rien n'est sûr....
- Donc dans le pire des cas, on ne termine qu'un seul processus par étape (équivalent à l'exécution séquentielle).
- Et dans le meilleur cas, tout se termine à la première ou seconde étape.

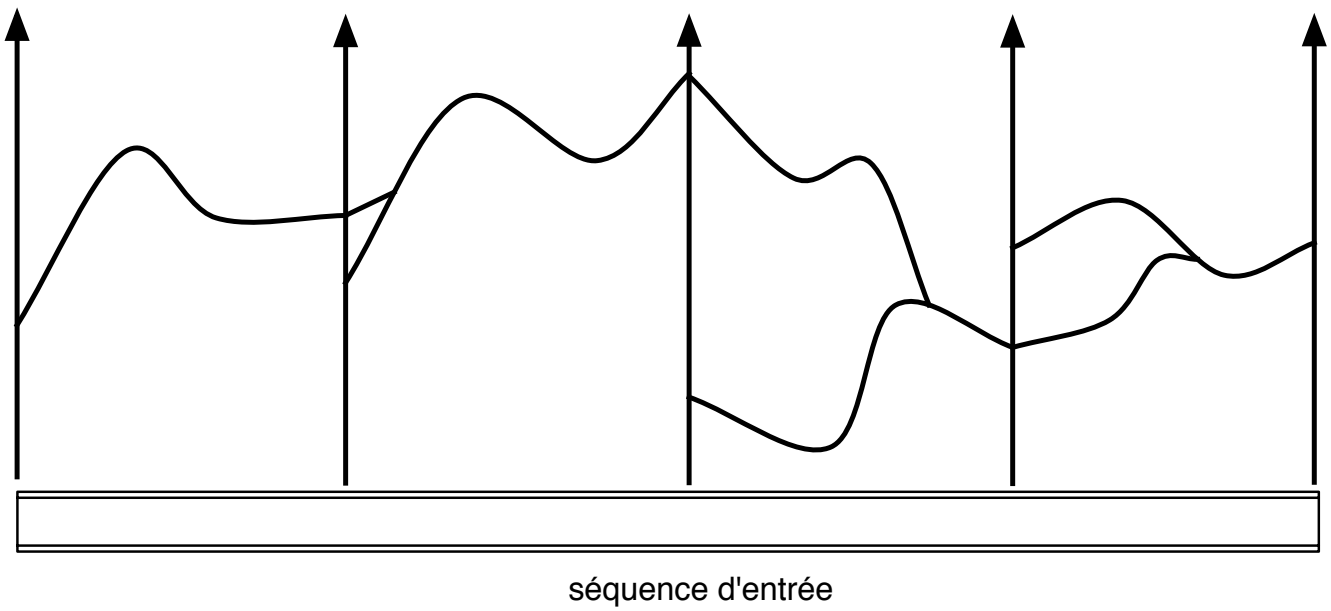

Figure 3: TPS sur 4 processeurs terminant en deux phases

Exemple: un cache LRU

Etape 1

1	2	1	3	4	3	6	7	2	1	2	6	9	3	3	6	4	2	3	1	7	2	7	4

1	2	1	3	4	3	6	7	2	1	2	6	9	3	3	6	4	2	3	1	7	2	7	4
	1	2	1	3	4	3	6		2	1	2	6	9	9	3		4	2	3	1	7	2	7
			2	1	1	4	3				1	2	6	6	9			4	2	3	1	1	2
				2	2	1	4					1	2	2	2				4	2	3	3	1

Etape 2

1	2	1	3	4	3	6	7	2	1	2	6	9	3	3	6	4	2	3	1	7	2	7	4

								2	1	2	6	9				4	2	3	1				
								7	2	1	2	6				9	4	2	3				
								6	7	7	1	2				6	9	4	2				
								3	6	6	7	1				2	6	9	4				

- Le processus 2 couple à la date 5
- Le processus 3 couple à la date 4

Partie 2 : Propriétés Trajectorielles

- Peut on trouver des points X_t sans calculer tous les points de la trajectoire entre 0 et t ?
- Oui dans certains cas très particuliers....
- Dans ce cas, on calcule les points à l'étape 1 et on effectue en parallèle les simulations entre les points successifs.

Exemple

- On considère un multiplexeur de cellules ATM. Le débit de sortie est C. Il y a K voies en entrées. Il y a un buffer de taille B.
- On représente la superposition des entrées par la séquence (N_t, D_t). N_t est le nombre d'arrivées par unité de temps sur l'intervalle $\sum_{i<t} D_i$ et $D_t + \sum_{i<t} D_i$.
- Comment évolue le système ?
- Entre $\sum_{i<t} D_i$ et $D_t + \sum_{i<t} D_i$, il entre N_t cellules par unité de temps (si il y a de la place) et il en sort C par unité de temps si il y en a assez.

- Soit X_t le nombre de clients à la fin de la période t. $X_0 = 0$.
- Equation d'évolution : signe de $C - N_t$
- Si $C > N_t$, la file se vide

$$X_{t+1} = min(0, X_t - D_t * (C - N_t))$$

- Si $C = N_t$, la file est constante

$$X_{t+1} = X_t$$

- Si $C < N_t$, la file se remplit

$$X_{t+1} = max(B, X_t + D_t * (N_t - C))$$

Borne

- Si $C > N_t$ et $D_t * (C - N_t) > B)$ alors $X_{t+1} = 0$ quelque soit X_t.
- Si $C < N_t$ et $D_t * (N_t - C) > B)$ alors $X_{t+1} = B$ quelque soit X_t.
- Donc il suffit d'analyser la séquence des (N_t, D_t) pour trouver des points où X_t vaut 0 ou B et de commencer les simulations en parallèle par ces points.

Part 3: improving the efficiency of TPS

- Compute Bounds instead of Estimators
- Simulation = black box used to build an output sequence from an input sequence and hidden variables (initial states)
- Based on the comparison of sequences
- Monotonicity property of the model

Improving TPS - Second approach

- Problem: the number of guessed points may be small (small degree of parallelization)
- Solution Here: change the input sequence I_t into H_t such that:
- The points guessed with I_t are also guessed with H_t
- Some points guessed with H_t were not guessed with I_t
- The simulation output sequence with H_t is a bound of the simulation output sequence with I_t

Comparison of Sequences

- Several ordering possible for sequences.

  Definition 1 *Let I_1 and I_2 two sequences with length n, $I_1 \preceq_p I_2$ if and only if $I_1(t) \leq I_2(t)$ for all index t smaller than n.*

  Definition 2 *Let I_1 and I_2 two sequences with length n, $I_1 \preceq_{sum} I_2$ if and only if $\sum_{t=1}^{m} I_1(t) \leq \sum_{t=1}^{m} I_2(t)$ for all index m smaller than n.*

  Property 1 *Assume that the rewards are computed with function r. If the rewards are increasing with the sequence, then $I_1 \preceq_p I_2$ implies that $r(I_1) \leq r(I_2)$. Many rewards such as moments and tails of distribution are increasing.*

- strong relations with the Stochastic Comparison of random variables (see Stoyan, Kijima, Ross, Shantikumar)

Monotonicity

- A simulation model is defined as an operator on an input sequence.
- Let $\mathcal{M}$ be a simulation model. We denote by $\mathcal{M}(I)$ the output of model $\mathcal{M}$ when the input sequence is I.

  Definition 3 (is-monotone Model) *Let $\mathcal{M}$ be a simulation model, and let $\preceq_\alpha$ and $\preceq_\beta$ be two orderings on sequences, $\mathcal{M}$ is input-sequence monotone with orders $\preceq_\alpha$ and $\preceq_\beta$ if and only if for all sequences such that $I_1 \preceq_\alpha I_2$ then $\mathcal{M}(I_1) \preceq_\beta \mathcal{M}(I_2)$.*
- Preservation of the ordering.

Example:$\mathcal{M}_2$ [Batch/D/C/K queue]

- Consider a finite capacity (K) with C synchronous servers with arrivals before service
- Service time is constant and equal to one time slot.
- Arrivals: i.i.d. batch process and that
- The input of the model is the number of arrivals at time t (say $I(t)$) and the output of the model (say $X(t)$) is the queue size at time t.
- Hidden variable: size of the queue at time 0 equal to 0.
- Evolution equation:

$$X(t+1) = min(K, max(0, X(t) - C + I(t))).$$

- Model $\mathcal{M}_2$ is not is-monotone with order $\preceq_{sum}$ but it is monotone with order $\preceq_p$.

Strategy to modify the input sequence

- Based on the equation of the model.
- For $\mathcal{M}_2$: guessed points: starvation and overflow
- New sequence: add more arrivals when the arrivals allow to guess that we are closed to overflow. New detection of overflow. More parallel processes.
- The modified sequence is bigger with order $\preceq_p$.
- The model is $\preceq_p$ monotone. Thus the output sequence with the modified input is also a bound of the original result.
- A more parallel upper bound of the original model.

Improving TPS - First approach

- Problem: avoid the corrections in the optimistic approach.
- Solution Proposed: compute bounds if the system is monotone.
- Upper bounds: if the initial point is bigger than the exact point, do not correct
- Lower Bound : if the initial point is smaller than the exact point, do not correct
- The simulation output sequence with is a bound of the exact simulation output sequence.
- less corrections: less iteration to couple.

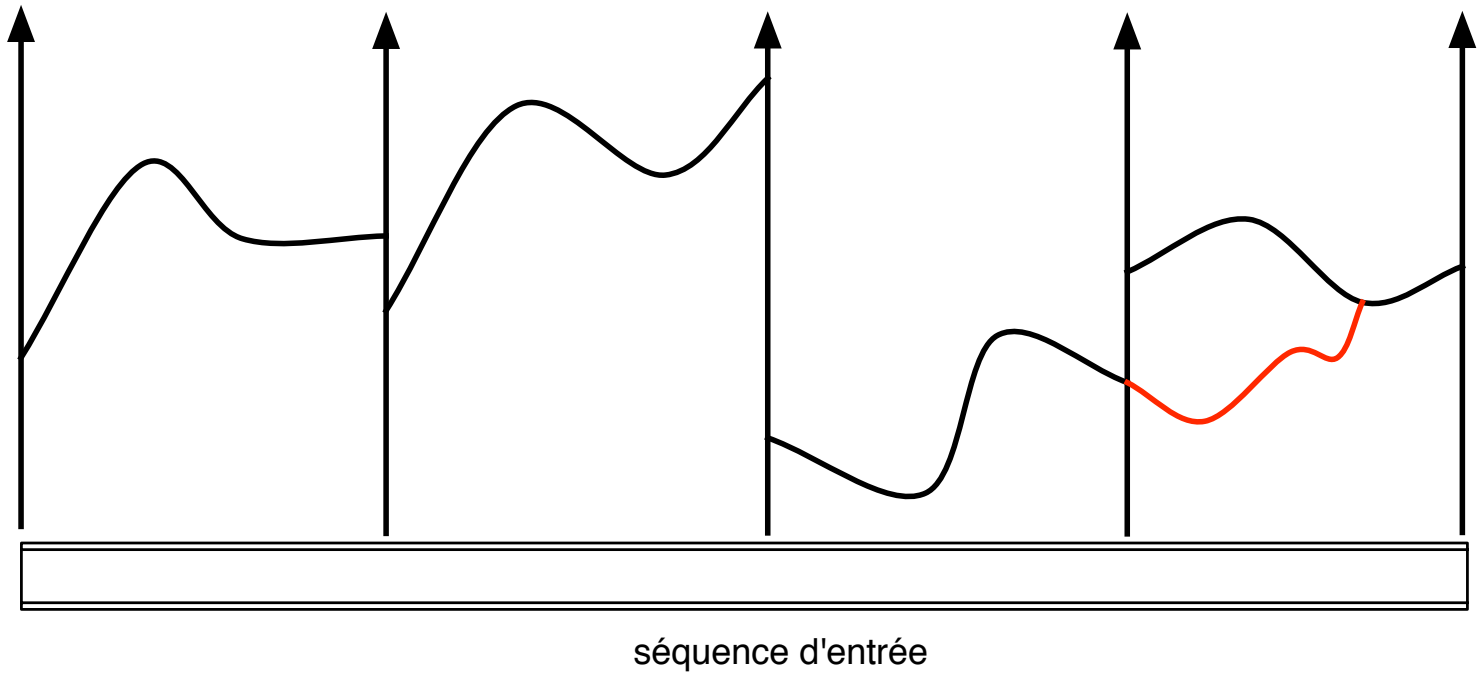

Figure 4: Simulation Temps parallèle sur 4 processeurs , phase 2

Après la phase 1

- Intervalle 1 : Exact
- Intervalle 2 : Borne Inf prouvée (monotonie)
- Intervalle 3 : Borne Inf prouvée (monotonie)
- Intervalle 4 : Supérieure à la borne inf : donc on ne sait pas. On fait une seconde phase pour l'intervalle 4.

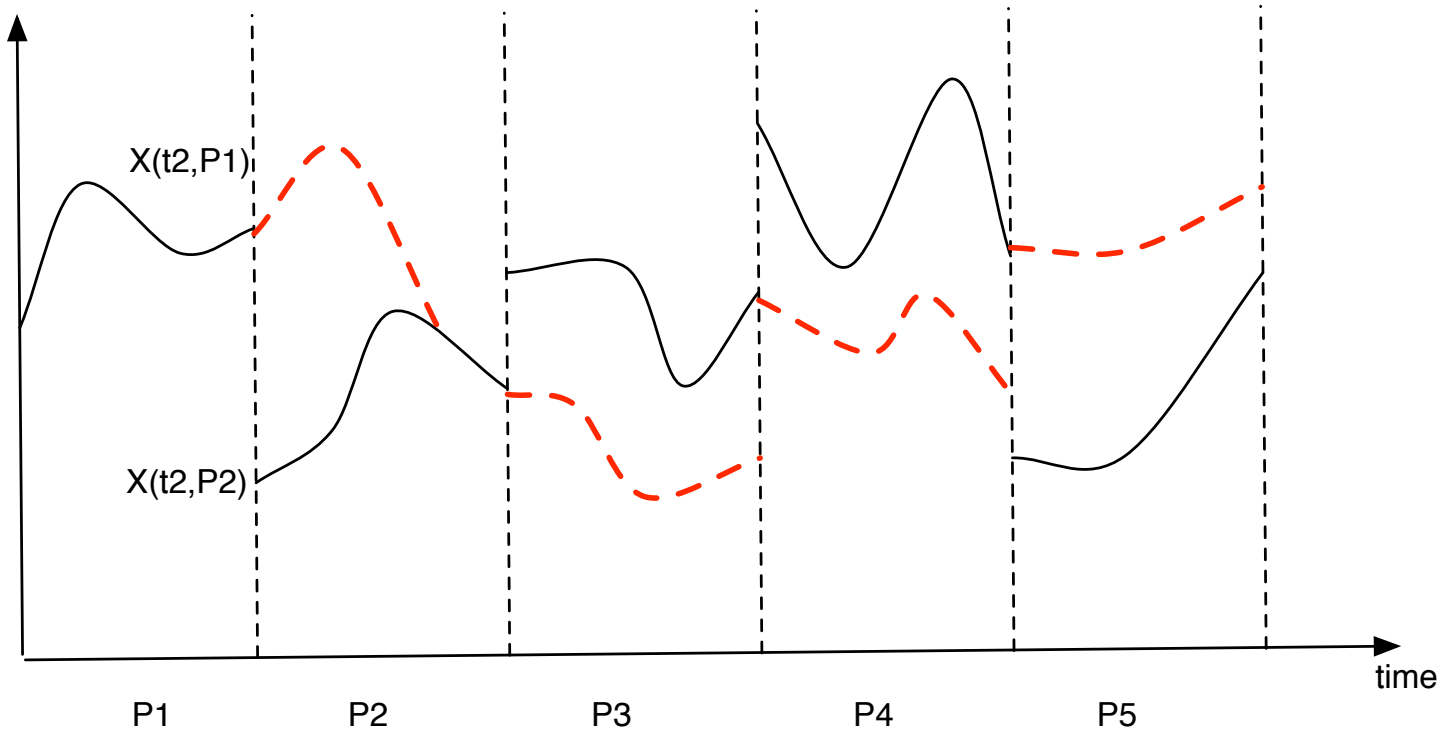

Figure 5: TPS sur 5 processeurs, convergence en deux phases pour une borne supérieure, phase 1 noir, phase 2 rouge

Quelques idées à creuser

- Calculs spéculatifs: si un processeur a une trajectoire qui a couplé mais ne sait pas si la trajectoire est valide, il peut faire une nouvelle trajectoire depuis un autre point pour avoir une alternative pour construire la borne.
- Affecter plusieurs processeurs inutilisés à un seul intervalle pour avoir plusieurs trajectoires
- Construire une borne sup et une borne inf en même temps.

Exemples

- Switch optique ROMEO
- Web Server

Conclusion

- IMPLEMENTATION
- PROJET ANR A DEPOSER... Parallelisme et Simulation (COSINUS)