Compte rendu TP :

Conception et mise en oeuvre d'un outil de simulation à événements discrets de systèmes hybrides

M2-IARF 2021/2022

MEKKANI Zakariae

NADDAM Reda

I-Objectif:

L’objectif de ce tp est de réaliser un outil de simulation a événements discrets de systèmes hybrides

II-Conception:

La réalisation de ce projet est faite par adoption de la méthodologie de programmation orientée objets, et donc par conséquent il est indispensable d’établir un diagramme de classe.

Le choix des classes est évidemment pris afin de répondre aux besoins.

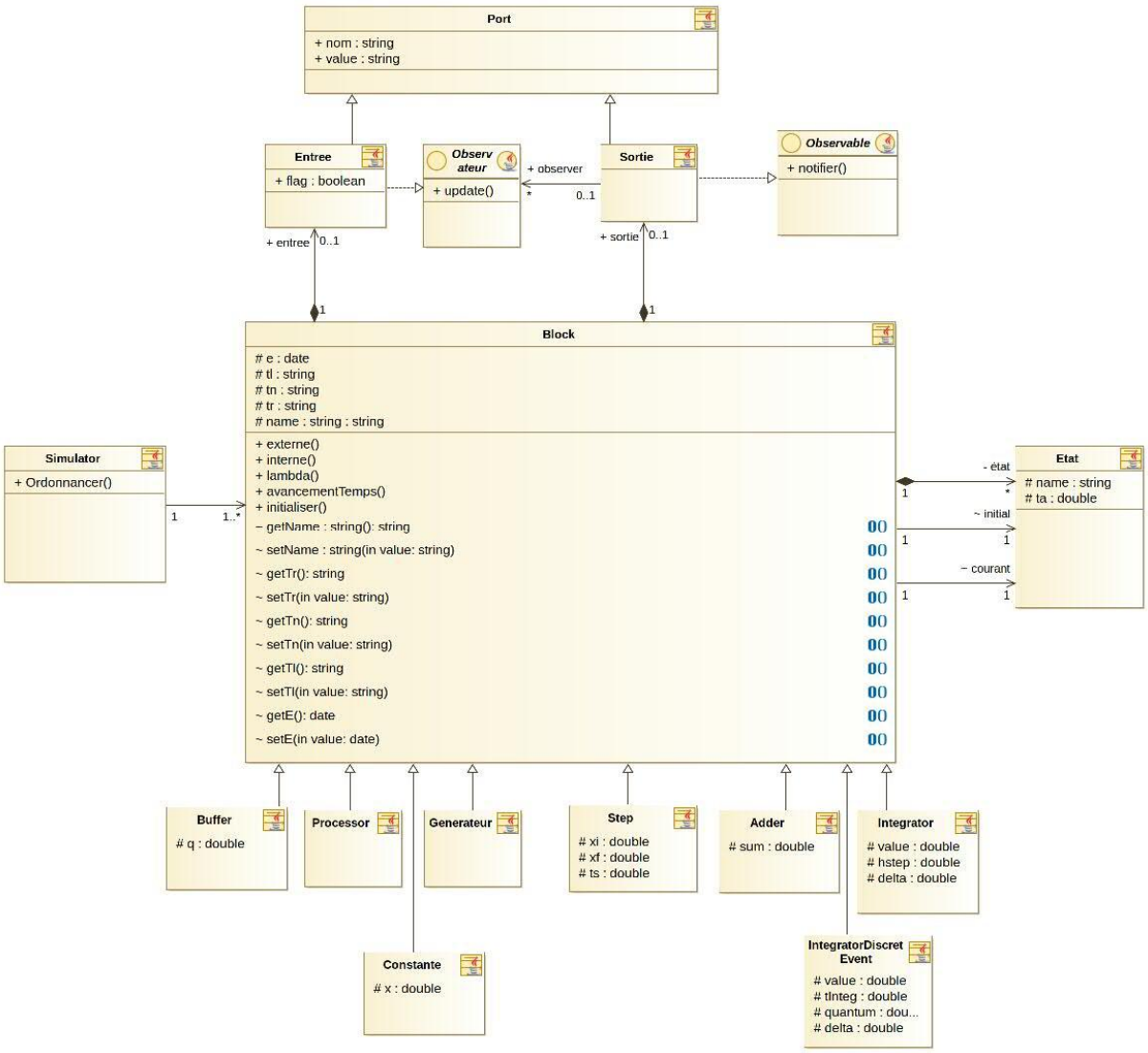

Figure 1 : Diagramme de classes

1-Exemple du gen-buf-proc:

Le modèle de simulation du mécanisme d'ordonnancement des tâches dans un système temps réel est le premier modèle mis en œuvre. Les machines à états du générateur, buffer et processeur (cf.fig 2) ont été données et explicitées pendant le premier cours.

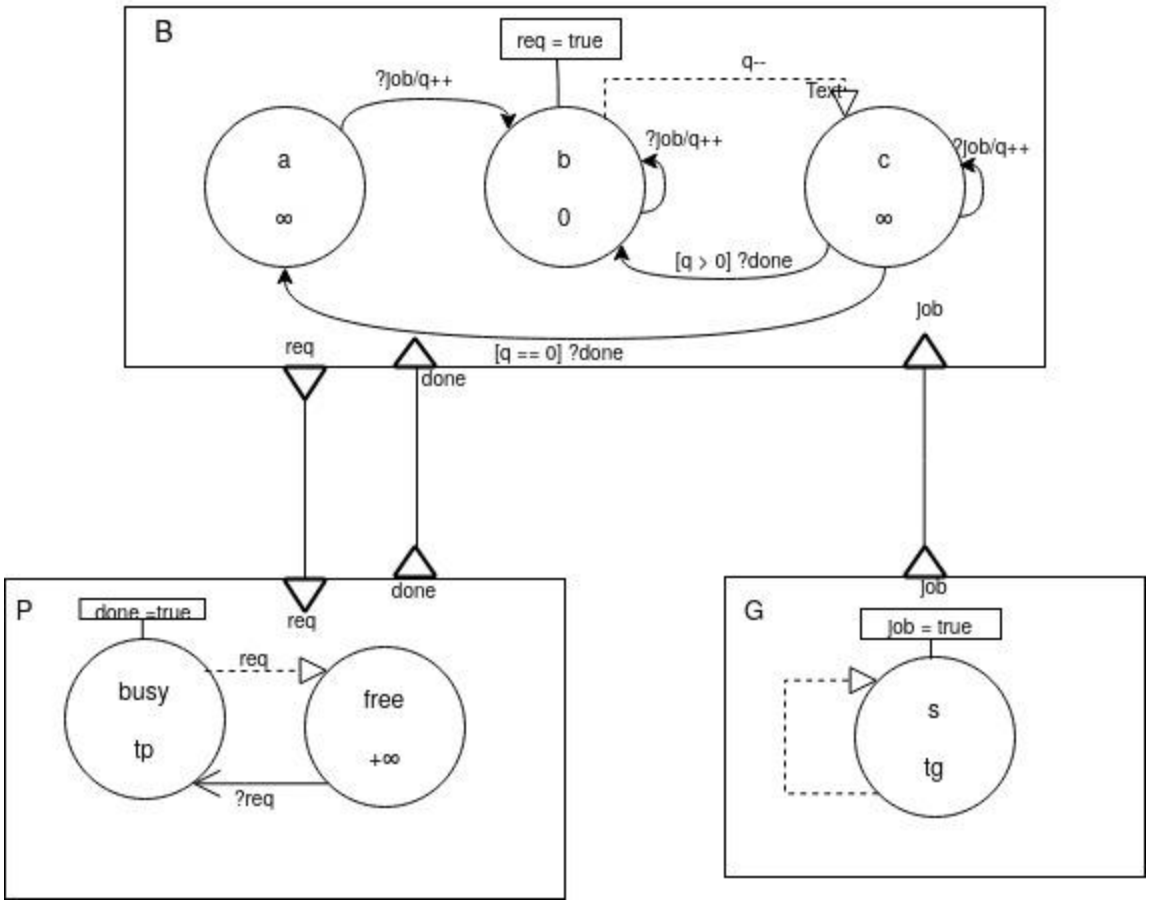

Figure 2 : Machines à états du générateur, processeur, buffer.

2-Exemple ODE du premier ordre:

Modélisation de la consigne:

Pour modéliser le modèle Stepper Adder on est parti sur deux composants atomique le premier est le composant du stepper qui contient trois états de transition “init” ,” wait” et “end“ avec une sortie. Ensuite on a conçu le composant Adder qui contient deux états de transition “wait” et “addition” avec quatre entrées et une sortie , les sorties des steppers sont liées avec les entrées de l’Adder.

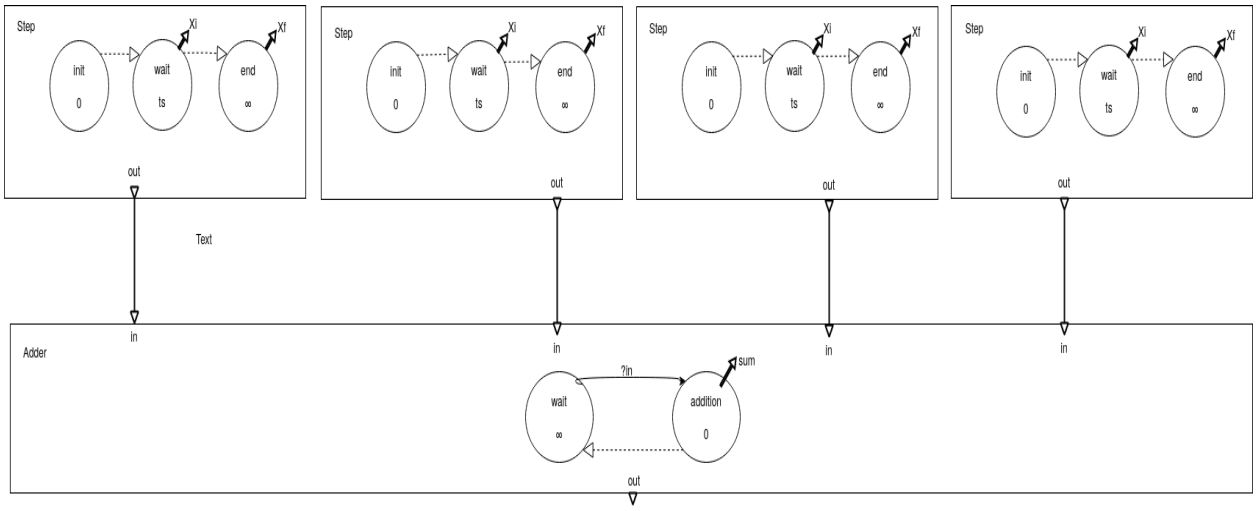

Figure 2 : Machines à états du stepper et adder.

Modélisation d'un intégrateur à temps discret:

Pour modéliser l'intégrateur a temps discret on a rajouté à la structure précédente le composant intégrateur temps discret qui contient deux états de transition “init” et “integ” avec une entrée et une sortie et on a lié la sortie de l’adder avec l’entrée de l'intégrateur .

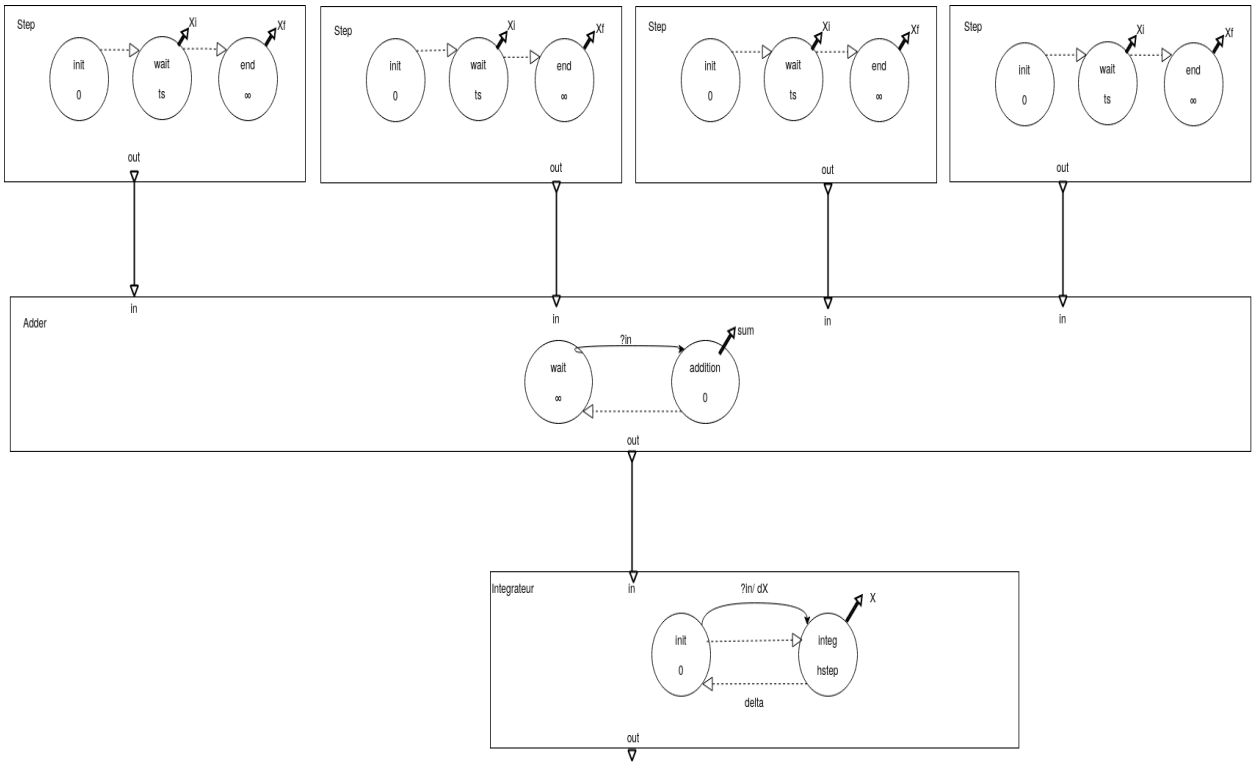

Figure 3 : Machines à états intégrateur à temps discret.

Modélisation d'un intégrateur à événement discret:

Comme il est décrit dans l'énoncé l'intégrateur à temps discret devrait à chaque écoulement d'une durée calculer la valeur l'intégrale de dérivée récupérée en entrée puis ajouter ou bien soustraire un quantum en fonction du signe de la cette dérivée.

Pour réaliser ceci nous avons créé notre intégrateur à événements discrets avec deux états; Un état initial chargé de récupérer la dérivée en entre et la transmettre cette valeur immédiatement à l'aide d'une transition externe. Un deuxième état a été défini avec un temps d'intégration *tInteg* permet de calculer l'intégrale de la valeur récupérée en entrée et incrémenter ou décrémenter un quantum à ce résultat à chaque écoulement de *tInteg.*

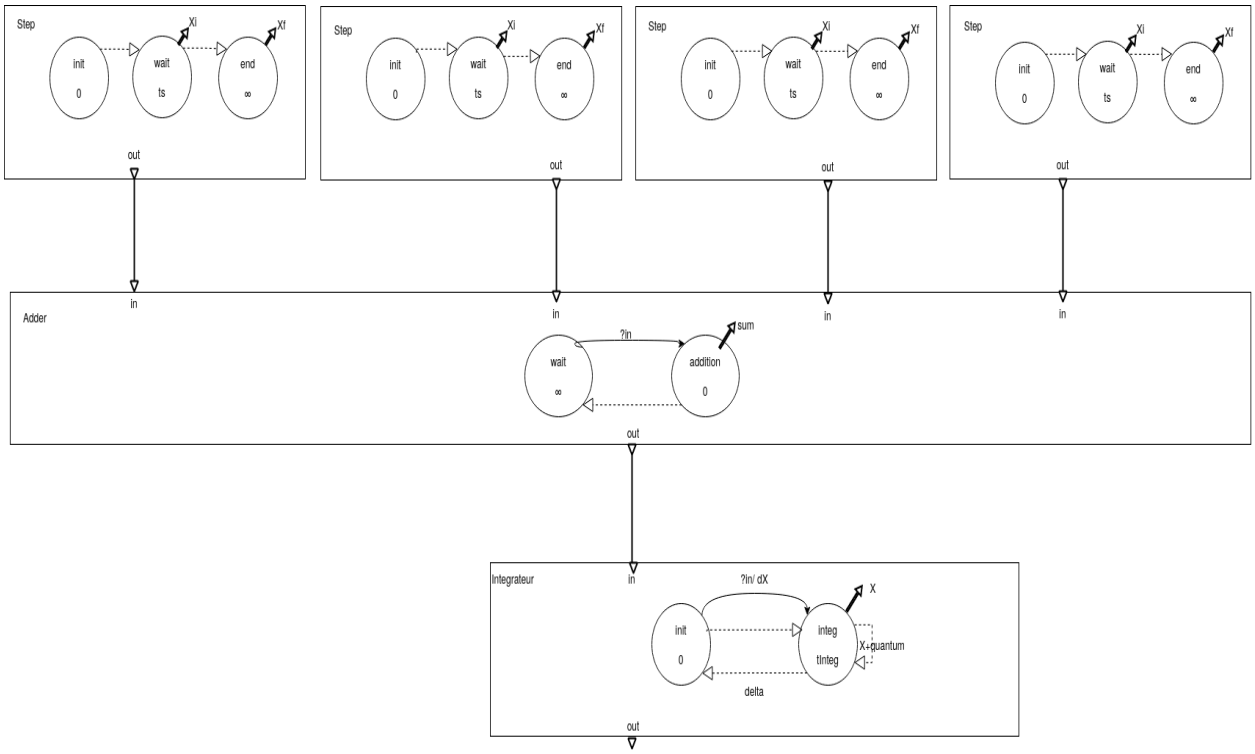

Figure 4 : Machines à états intégrateur à événement discret.

3- ODE du second ordre & Bouncing ball:

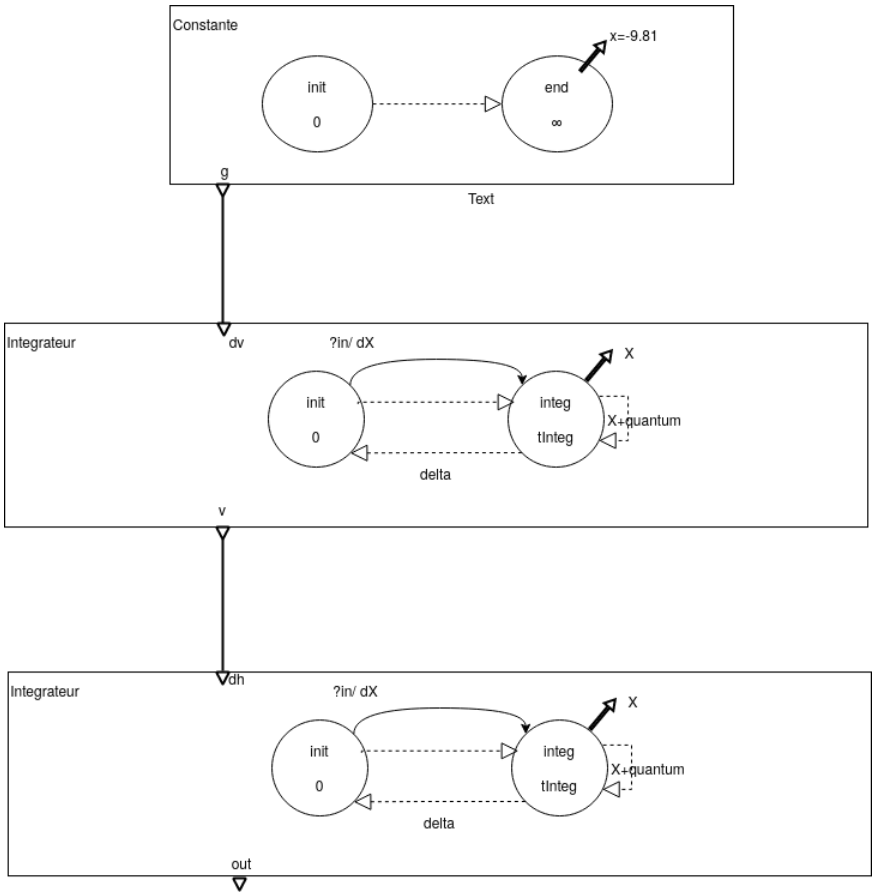

Figure 5 : Machines à états ODE second ordre.

Cette manipulation permet de faire une simulation des systèmes à temps continu en réalisant une double intégration à l'aide de l'intégrateur à événements discrets.
Nous avons modélisé le composant *Constante* avec deux états *init* et *end* liés par une transition interne.

III-Code:

Après l'étude du problème et la conception des machines à état ainsi que le diagramme de classe nous avons passé à la partie codage pour cela nous avons utilisé le langage orienté objets JAVA.

“ Vous trouvez le code ci-joint avec ce fichier sous format Zip”.

IV-Resultat:

1-Exemple du gen-buf-proc:

Après l'exécution du programme sur le modele generateur buffer processeur (la fonction *genBufProc*() dans le code JAVA) et en fixant la valeur de *Tfinal* sur 6 on obtient le résultat représenté sur la figure 3 :

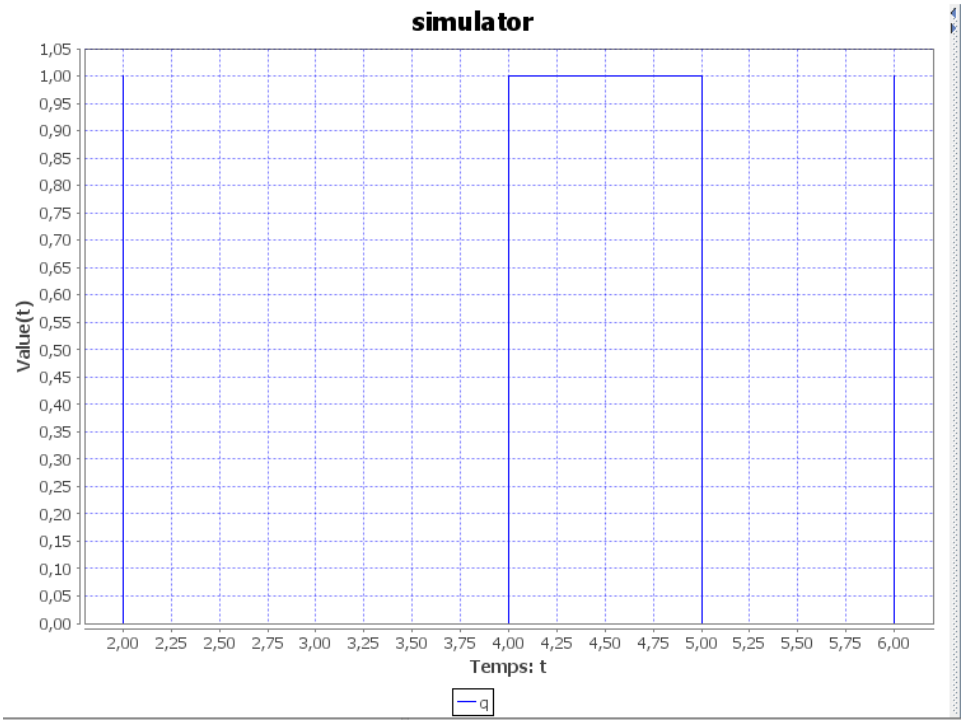

Figure 3 : Résultat de simulation Générateur-Buffer-Processeur

Cette fois on a fixé la valeur de Tfin sur 20 ce qui nous a donné le résultat représenté sur la figure 4 :

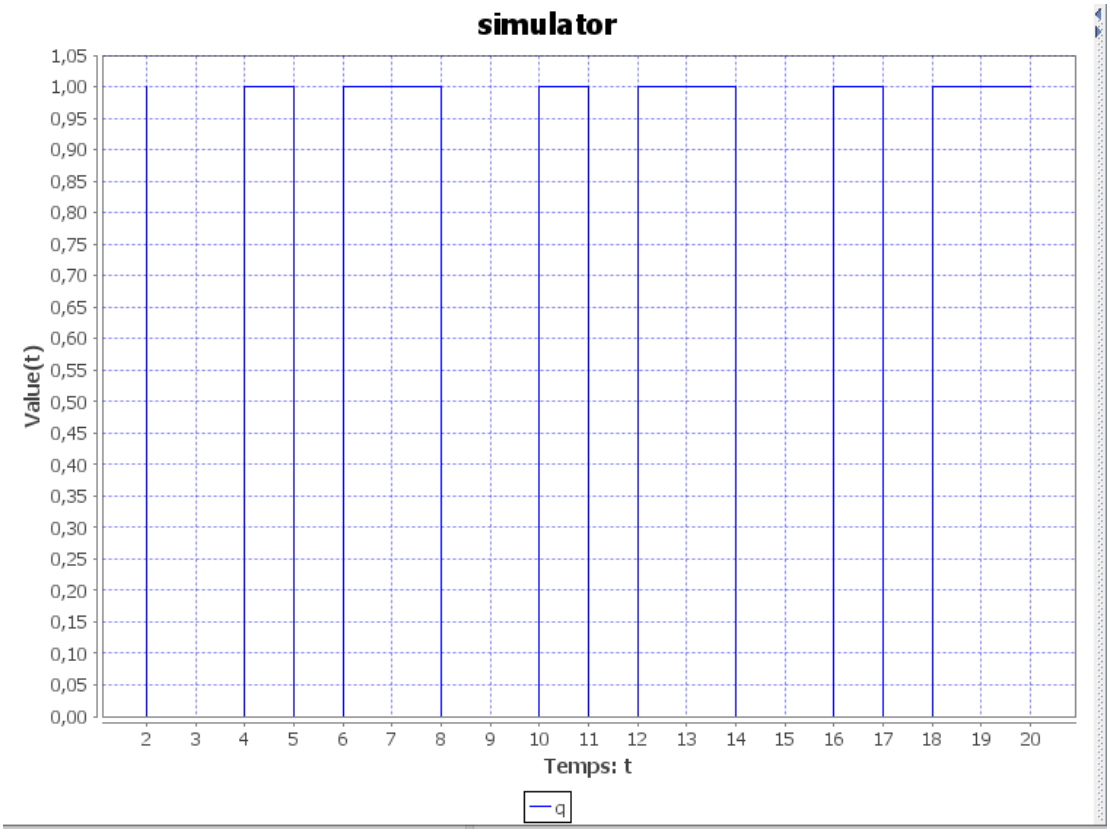

Figure 4 : Résultat de simulation Générateur-Buffer-Processeur

2-Exemple ODE du premier ordre:

Pour le modèle Steppers Adder on a fixé Tfin sur 2 et on a lancé la simulation à l'aide de la fonction *Add_Steps*() ce qui nous a donné le résultat représenté sur la figure 5 :

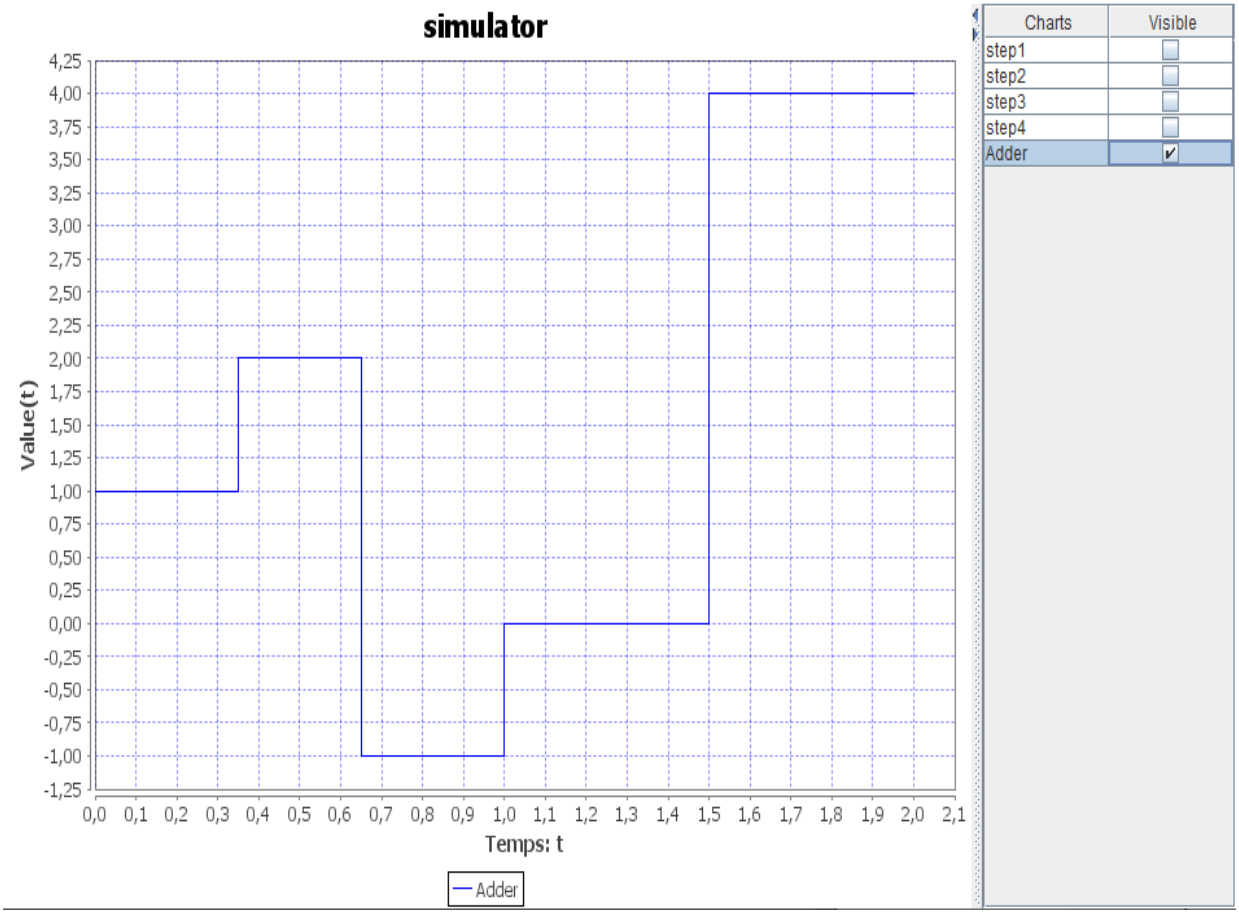

Figure 5 :Résultat de simulation Stepper-Adder

Ensuite on a créé la fonction *discretTimeIntegration*() où on gardé l'ancien modèle de steppers Adder mais cette fois on a ajouté le composant intégrateur à temps discret donc on a lié la sortie de l'adder avec l'entrée de l'intégrateur et on a fixé Tfin sur 3 ce qui nous a donnée le résultat représenté sur la figure 6 :

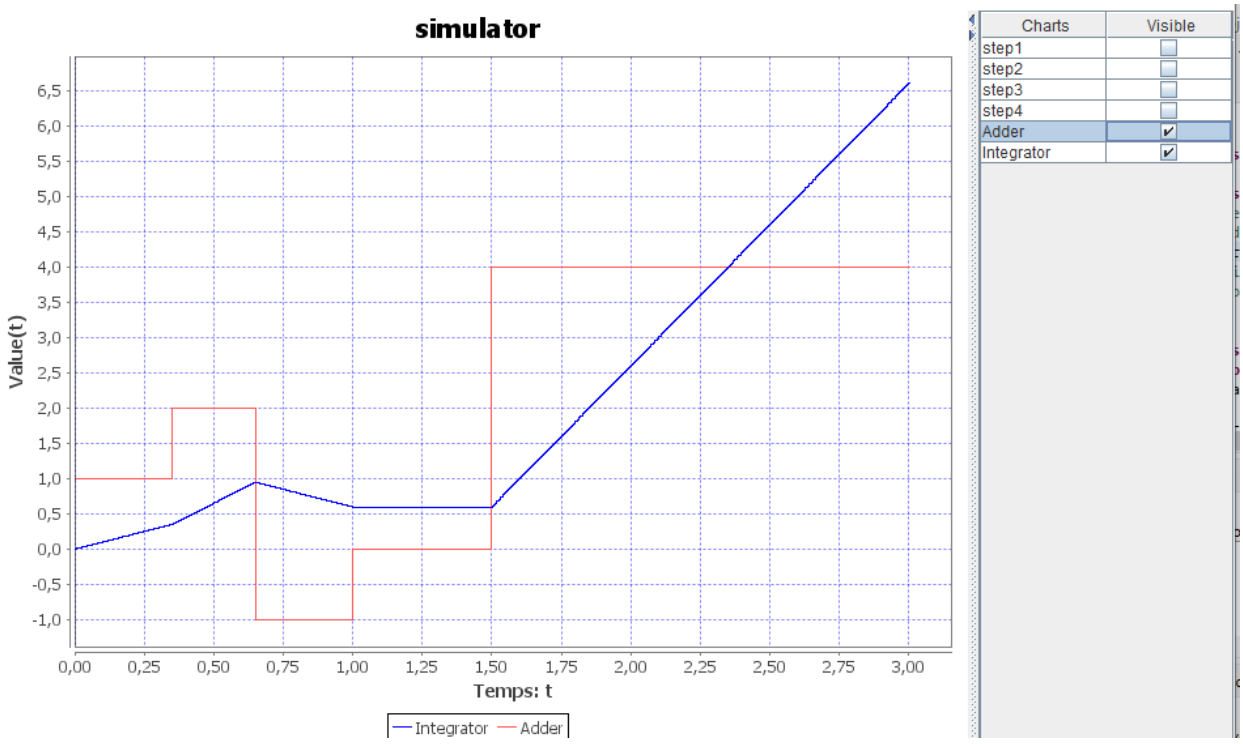

Figure 6 : Résultat après l'ajout de l'intégrateur à temps discret

Pour afficher le résultat de simulation de l'intégrateur à événement discret on a créé une nouvelle fonction -discretEventIntegration()- où on a rajouté cette fois le composant de l'intégrateur à événement discret avec les steppers et l'adder en fixant Tfin à 3 ce qui nous a donnée le résultat représenté dans la figure 7 :

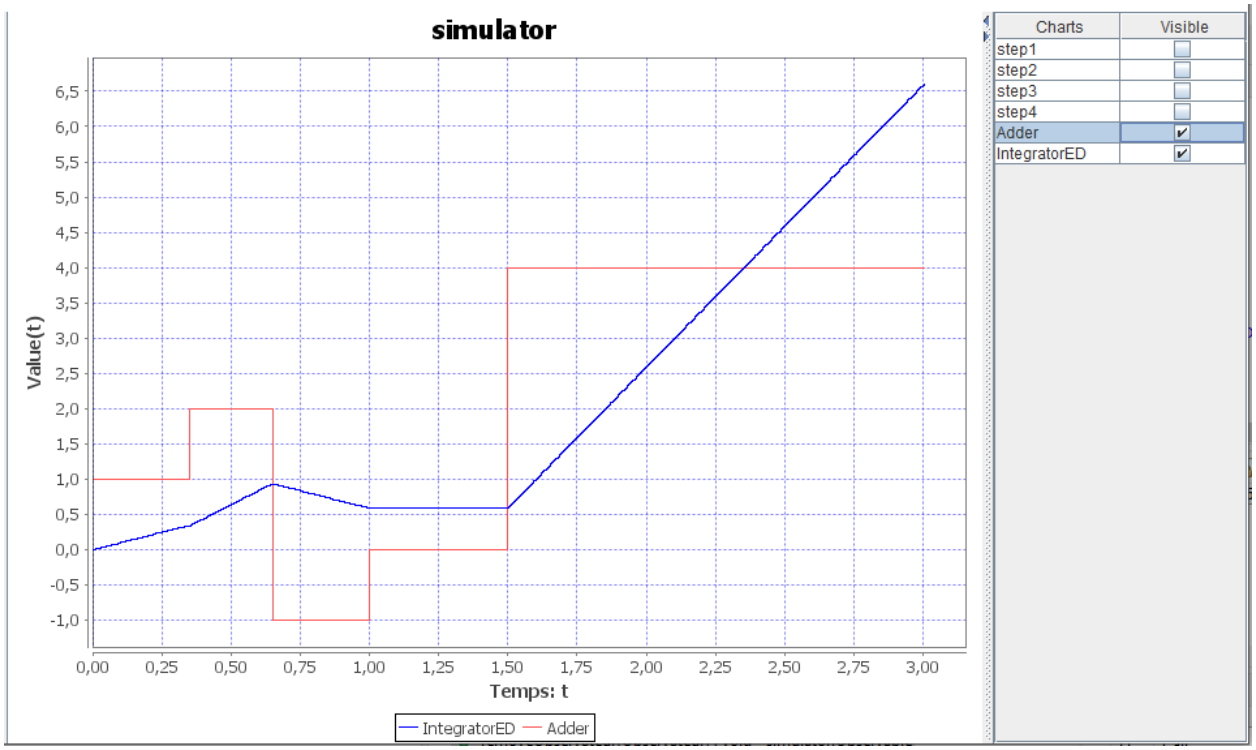

Figure 7 :Résultat après l'ajout de l'intégrateur a elements discret

Le résultat de la figure 8 représente la même simulation de l'intégrateur à événement discret mais avec un quantum égale à 0.1

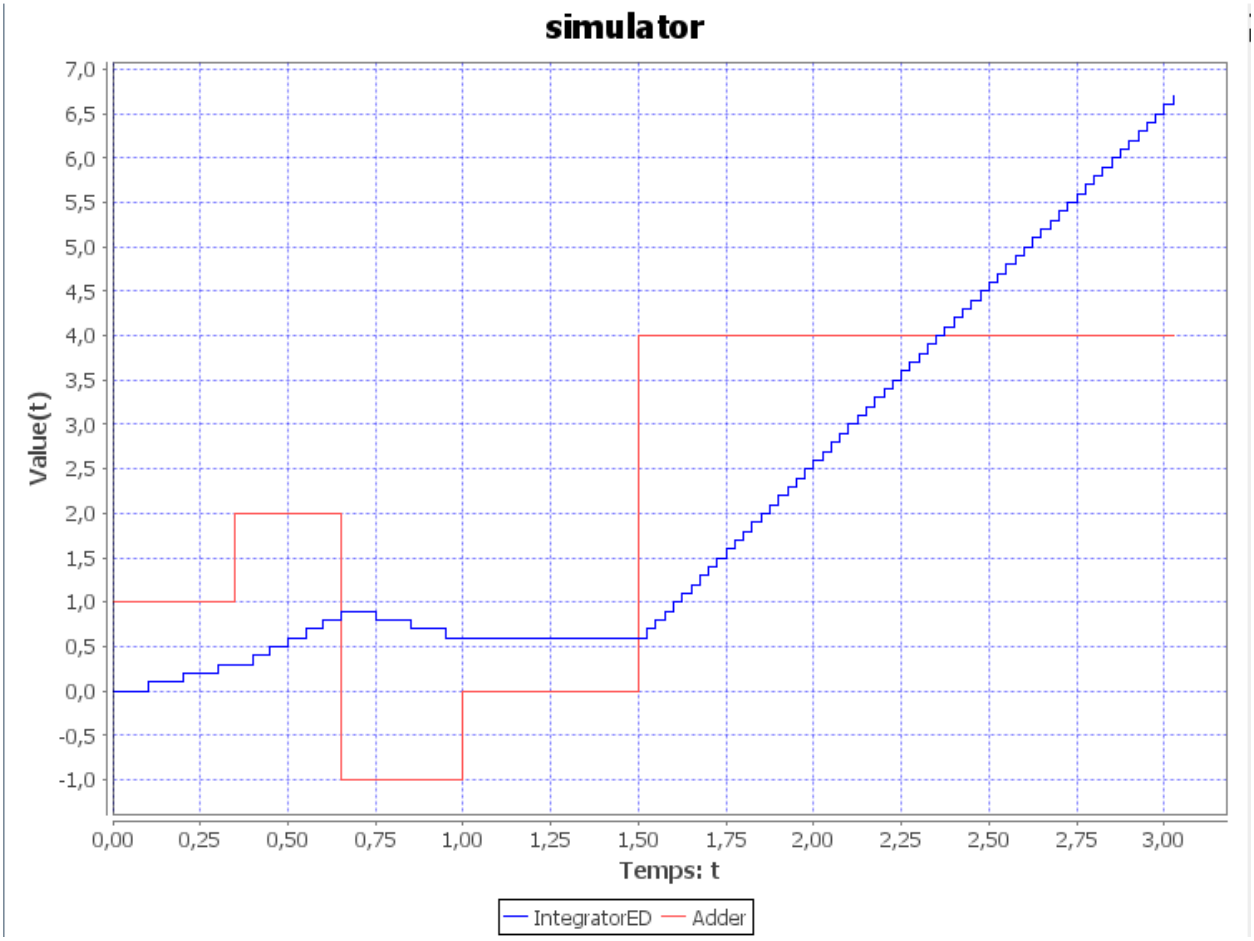

Figure 8: Résultat après l'ajout de l'intégrateur à éléments discret

3- ODE du second ordre & Bouncing ball:

Après l'exécution de la fonction de simulation de l'exemple ODE du second ordre -constantSecondOrdreIntegration()- ou on a relier le composant constat avec deux intégrateurs éléments discrets avec Tfin toujours fixé sur 3 on a obtenu le résultat représenté dans la fig 9.

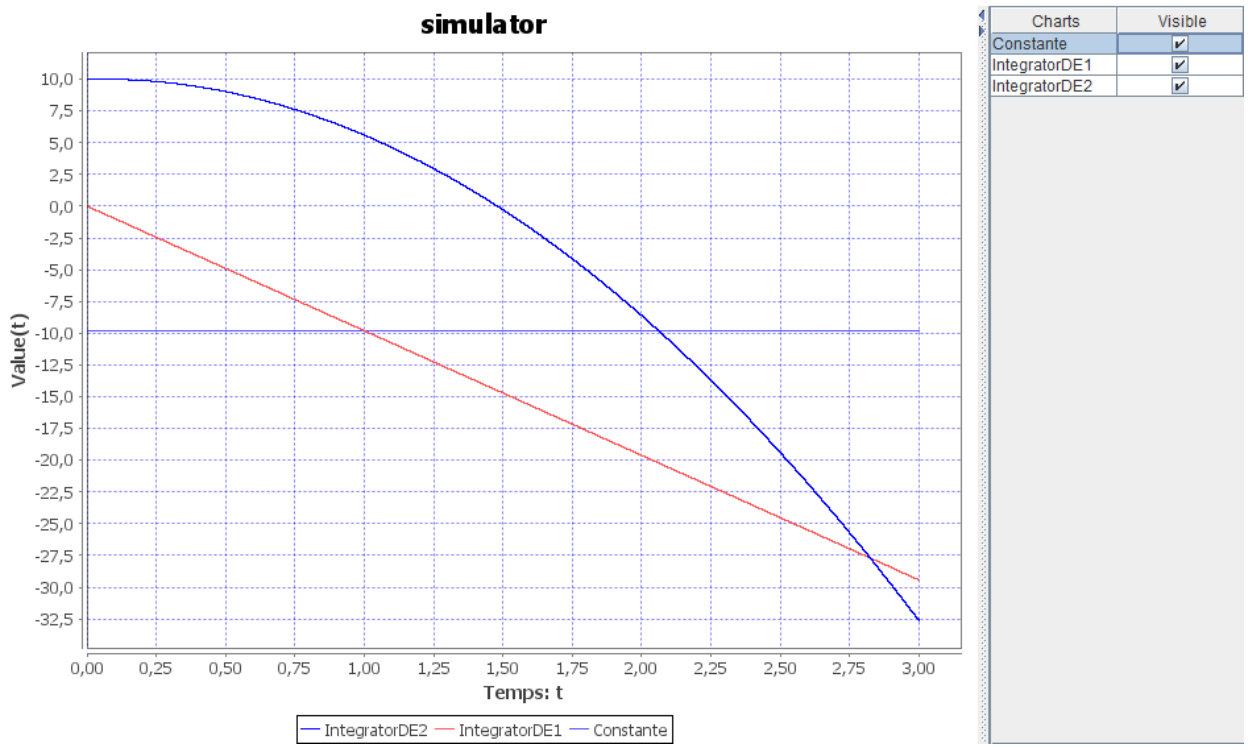

Figure 9:Résultat de simulation de l'exemple ODE seconde ordre

Enfin on a fait la simulation de l'exemple du Bouncing Ball a l'aide de la fonction constantSecondOrdreIntegration() avec Tfin = 10 d'où on a eu le résultat présenté dans la figure 10 .

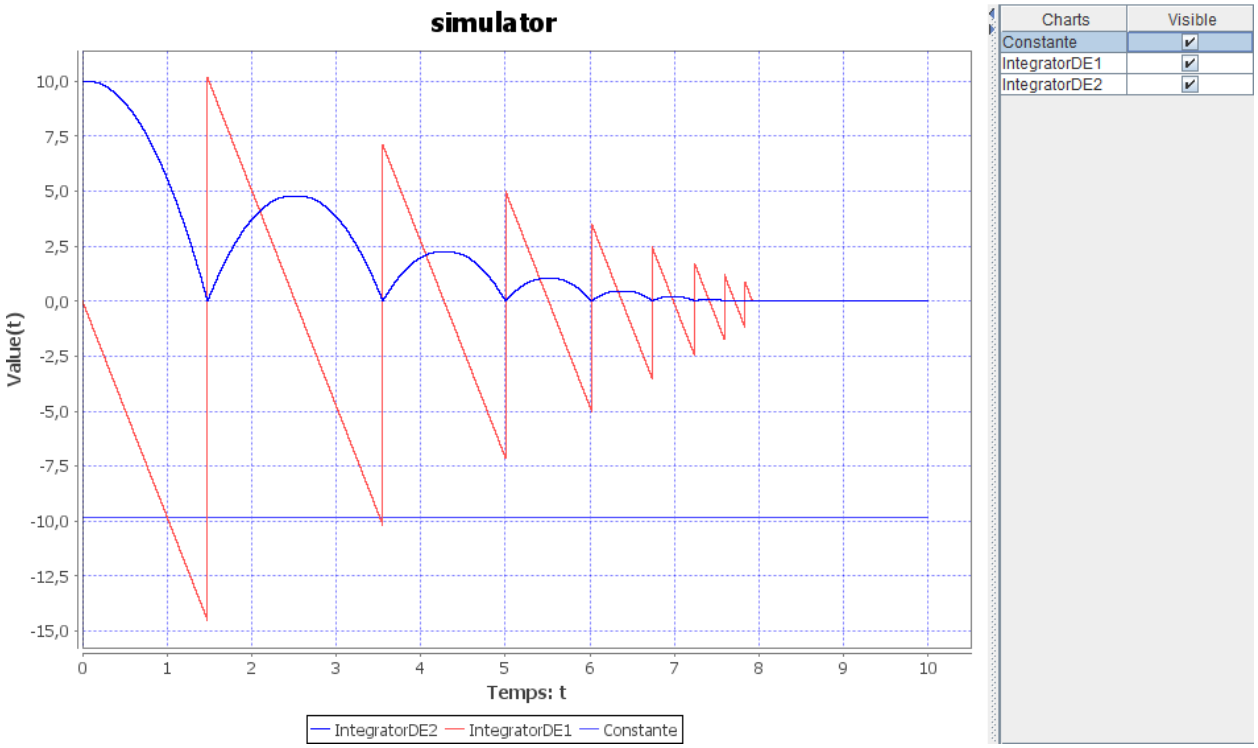

*Figure 10:*Résultat de simulation de l'exemple du Bouncing ball

Conclusion:

Certes il existe une variation d'outils et de logiciels pour faire la modélisation et la simulation des systèmes temps réels et donc nous permettre à apprendre sur ce genre de systèmes. Par contre ces travaux pratiques nous ont permis de décortiquer un système de mise en pratique de nos compétences de conception et de programmation orientée objets acquises.