

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

CENTRE RÉGIONAL ASSOCIÉ DE NOUVELLE-AQUITAINE

MÉMOIRE

présenté en vue d'obtenir

le DIPLÔME d'INGÉNIEUR·E CNAM

SPÉCIALITÉ : Informatique

Parcours Big Data et Intelligence Artificielle

EN PARTENARIAT AVEC l'Université de Poitiers

par

AUVRAY Marc-Antoine

**Intégration de temps d'exécution probabilistes dans
l'interface utilisateur d'un outil simulateur
d'ordonnancement temps-réel**

Soutenu le 09/05/2023

JURY

PRÉSIDENT·E :

MEMBRES :

Remerciements

Les travaux présentés dans ce mémoire se sont déroulés au sein d'une équipe de l'Institut National de Recherche en Informatique et Automatique, avec l'accompagnement de Monsieur Kevin Zagalo.

Tout d'abord, je souhaite remercier chaleureusement Monsieur Kevin Zagalo doctorant que j'ai pu assister. Il m'a procuré la très grande partie des moyens nécessaires au bon déroulement de mon travail. Je le remercie sincèrement de l'attention particulière qu'il a portée sur mes travaux et de la confiance qu'il m'a donnée. Sa bienveillance et son soutien constant m'ont permis de me sentir en confiance et de me concentrer pleinement sur mes tâches. Je le remercie pour son suivi et le temps qu'il a consacré à mon travail : il m'a notamment enseigné le fonctionnement de plusieurs algorithmes de *machine learning* et la programmation orientée objet et m'a sensibilisé aux bases de la gestion de projet.

Je voudrais aussi remercier Monsieur Slim Ben Amor, Madame Yasmina Abdeddaïm, Madame Chiara Daini et Monsieur Marwan Wehaiba El Khazen : l'intérêt, leurs compétences et leurs précieux conseils ont grandement contribué à l'enrichissement de mes connaissances. Je leur suis très reconnaissant du temps qu'ils m'ont accordé pour répondre à toutes mes interrogations.

Mes remerciements vont enfin à Monsieur Ismail Hawaila, Madame Chiara Daini, Madame Yasmina Abdeddaïm, Monsieur Slim Ben-Amor, Monsieur Avner Bar-Hen, Monsieur Mohamed Amine Khelassi, Monsieur Marwan Wehaiba El Khazen et Monsieur Kevin Zagalo pour leur accueil très sympathique.

Liste des sigles

- IA : Intelligence Artificielle. IA : Intelligence Artificielle.
- INRIA : Institut National de Recherche en Informatique et Automatique.
- IRIA : Institut de Recherche en Informatique et Automatique.
- PX4 : *Pixhawk* 4.
- SCP : Système Cyber-Physique.
- EPST : Etablissement Public à caractère Scientifique et Technologique.
- WCET : *Worst Case Execution Time* (temps d'exécution pire cas).
- CPU : *Central Proccessing Unit* (processeur ou microprocesseur principal d'un ordinateur).
- COP : Contrat d'Objectifs de Performance.
- ETM : *Execution Time Model* (modèle de temps d'exécution).
- pET : *Probabilistic Execution Time* (temps d'exécution probabiliste).
- ACET : *Average Case Execution Time* (Temps d'exécution en cas moyen).

Glossaire

- Temps-réel : comportement d'un système informatique dont le fonctionnement est assujéti à l'évolution dynamique d'un procédé industriel connecté à lui.
- Système temps-réel : système capable de contrôler (ou piloter) un procédé physique à une vitesse adaptée à l'évolution du procédé contrôlé.
- Ordonnancement : organisation des traitements, leurs enchainements entre eux, leur planification, la synchronisation des travaux et leur exécution.
- Ordonnanceur : composant du noyau du système choisissant l'ordre d'exécution des processus sur les processeurs d'un ordinateur.
- Autopilote : pilote automatique ou en partie automatique.
- Microcontrôleur : ordinateur présent dans un seul circuit intégré qui est dédié à effectuer une tâche et exécuter une application spécifique.
- Systèmes embarqués temps-réel : systèmes informatiques qui ont des contraintes de temps très strictes, conçus pour des applications spécifiques à un appareil et qui ont pour but de garantir une réponse fiable et prévisible aux exigences de temps.
- Ordonnancement temps-réel : technique d'ordonnancement de tâches dans les systèmes embarqués temps-réel.
- *SimSo* : un outil simulateur d'ordonnancement temps-réel qui permet de simuler des ordonnancements temps-réel prenant en entrée des paramètres tels que la politique d'ordonnancement, le type de processeur et un ensemble de tâches choisis par l'utilisateur.

Introduction

Ce mémoire de deuxième année d'ingénierie en informatique parcours Big Data & Intelligence Artificielle en alternance à l'Institut National de Recherche en Informatique et Automatique est un rapport des travaux sur l'intégration de temps d'exécution probabilistes dans l'interface utilisateur d'un outil simulateur d'ordonnancement temps-réel nommé SimSo. Ce mémoire est divisé en trois parties.

La première partie de ce travail se concentre sur l'environnement dans lequel le projet a été mené, en présentant notamment l'INRIA, la structure de l'organisme ainsi que la présentation économique et juridique.

La deuxième partie de ce mémoire traite de la problématique en détaillant le contexte, le problème rencontré et les attentes du projet, ainsi que la planification générale du projet.

La troisième partie de ce mémoire se concentre sur la recherche de solutions, en décrivant le choix de solution, les tests de validation, les tâches et actions à effectuer, ainsi que l'analyse des résultats obtenus. Plus particulièrement, cette partie se focalise sur l'utilisation de l'algorithme *K-means* pour l'analyse des temps d'exécution des tâches d'un système embarqué temps-réel, la familiarisation avec *SimSo*, un simulateur de système temps-réel, ainsi que l'implémentation de la possibilité d'ajouter des temps d'exécution probabilistes et d'extraire ces données dans des fichiers, avec prise en compte des nouvelles fonctionnalités.

Sommaire

Remerciements	2
Liste des sigles.....	3
Glossaire	4
Introduction	5
1. L'environnement.....	8
1.1. L'INRIA.....	8
1.1.1. Introduction	8
1.1.2. Structure	9
1.1.3. Présentation économique et juridique	10
1.1.4. Production.....	11
1.2. Equipe.....	12
1.2.1. Introduction	12
1.2.2. Travaux.....	12
2. Problématique	14
2.1. Contours	14
2.1.1. Préambule.....	14
2.1.2. Contexte.....	14
2.1.3. Problème rencontré.....	17
2.1.4. Attentes.....	17
2.2. Planification générale du projet	18
3. Méthodologie.....	18
3.1. Choix de solution.....	18
3.2. Tests de validation	19
3.3. Tâches et actions	19
3.4. Résultats obtenus	20
3.4.1. Utilisation de l'algorithme K-means pour l'analyse des temps d'exécution des tâches d'un système embarqué temps-réel	20
3.4.2. Familiarisation avec <i>SimSo</i>	21

3.4.3.	Implémentation de la possibilité d'ajouter des temps d'exécution probabilistes	25
3.4.4.	Implémentation de la possibilité d'extraire les données dans des fichiers, avec prise en compte des nouvelles fonctionnalités.....	34
3.4.5.	Ouvertures	36
Conclusion		41
Bibliographie		42
Liste des figures		43
Annexes		44
Résumé.....		45

1. L'environnement

1.1. L'INRIA

1.1.1. Introduction

Créée en 1967 à Rocquencourt dans les Yvelines en tant qu'IRIA, l'institut devient INRIA (Institut national de recherche en sciences et technologies du numérique) en 1979.

L'institut soutient la diversité des voies de l'innovation à travers notamment la création de startups technologiques et l'édition open source de logiciels. Elle est également fortement engagée dans le transfert de technologie par les partenariats noués avec des entreprises industrielles et aussi par l'intermédiaire de ses sociétés de technologie. (1)

Pour réaliser ses missions, l'INRIA s'appuie sur un modèle d'équipe-projet constitué de près de 200 équipes. Chaque équipe est composée d'une vingtaine de personnes, rassemblées autour d'un responsable scientifique et regroupant des profils complémentaires comme des doctorants, post-doctorants et ingénieurs. Les équipes-projets travaillent en collaboration avec des partenaires industriels et des institutions de recherche, tant en France qu'à l'étranger. Les projets de recherche et d'innovation menés par les équipes-projets sont évalués tous les quatre ans, ce qui assure une grande qualité de la recherche. En plus de ses cinq unités de recherche en France, l'INRIA a également des collaborations avec des partenaires dans d'autres régions du monde, renforçant ainsi son rayonnement international. L'INRIA est également fortement engagée dans la formation et l'accompagnement de startups technologiques, contribuant ainsi à l'innovation et au développement économique.

Moteur pour la recherche et l'innovation numérique, l'institut se voit confier différentes missions à travers la France et l'Europe tels que :

- Entreprendre des recherches fondamentales et appliquées ;
- Réaliser des systèmes expérimentaux ;
- Organiser des échanges scientifiques internationaux ;
- Assurer le transfert et la diffusion des connaissances et du savoir-faire ;
- Contribuer à la valorisation des résultats de recherches ;
- Contribuer, notamment par la formation, à des programmes de coopération avec des pays en voie de développement ;
- Effectuer des expertises scientifiques.

L'objectif principal est d'effectuer une recherche de haut niveau et d'en transmettre les résultats aux étudiants, au monde économique et aux partenaires scientifiques et industriels.

L'INRIA intervient dans cinq domaines de recherche, chaque domaine étant subdivisé en plusieurs thèmes :

- Santé, biologie et planète numériques ;
- Mathématiques appliquées, calcul et simulation ;
- Perception, cognition, interaction ;
- Réseaux, Systèmes et services, calcul distribué ;
- Algorithmique, programmation, logiciels et architectures.

1.1.2. Structure

Afin de répondre à de nombreux défis scientifiques et technologiques, près de 4000 chercheurs et ingénieurs explorent à l'Inria des voies nouvelles, que ce soit dans l'interdisciplinarité et en collaboration avec des partenaires industriels. Ce qui fait le cœur de l'INRIA est son modèle unique d'équipe-projet, en effet 200 équipes-projets qui constituent la cellule de base de la recherche au sein de l'institut. Constituée d'une vingtaine de personnes rassemblées autour d'un responsable scientifique, une équipe-projet agrège les talents nécessaires à la conduite d'un projet de recherche et d'innovation évalué tous les quatre ans. Généralement les équipes-projet sont composées de doctorants, post-doctorants et ingénieurs qui travaillent aux côtés de chercheurs confirmés.

L'INRIA s'est développé à travers cinq régions :

- En Ile-de-France, deux unités de recherche une à Paris et une à Saclay, ainsi que le siège qui se trouve à Rocquencourt ;
- En Bretagne, une unité de recherche à Rennes ; en Provence-Alpes-Côte d'Azur, une unité de recherche à Sophia Antipolis ;
- En Lorraine, une unité de recherche à Nancy ;
- En Rhône-Alpes, une unité de recherche près de Grenoble ;
- En Hauts de France, une unité de recherche à Lille.

En plus de ses cinq unités de recherche en France, l'INRIA a également mis en place des collaborations et des partenariats avec des institutions de recherche dans le monde entier. Cette stratégie de collaboration internationale permet à l'INRIA de renforcer son rayonnement scientifique et d'élargir son champ d'expertise en bénéficiant de perspectives et de connaissances diversifiées. L'INRIA participe activement à des programmes de coopération internationale, notamment avec les pays en voie de développement, en partageant son savoir-faire en matière de recherche et en contribuant à la formation de jeunes chercheurs. Cette ouverture internationale fait de l'INRIA un acteur clé de la recherche et de l'innovation

numérique à l'échelle mondiale, permettant ainsi de relever les défis scientifiques et technologiques de demain de manière collaborative et coopérative.

L'INRIA a lancé plusieurs projets de recherche en intelligence artificielle, tels que le projet Parietal qui se concentre sur la modélisation mathématique de la cognition humaine, le projet Sequel qui vise à développer des méthodes d'apprentissage automatique pour l'analyse de données massives, ou encore le projet Flowers qui étudie les systèmes cognitifs inspirés de la nature pour l'interaction homme-machine.

Ces projets ont permis à l'INRIA de collaborer avec des partenaires industriels et académiques à travers le monde, et de publier des résultats de recherche de premier plan dans des conférences et des revues scientifiques de renommée internationale. Les résultats de ces projets ont également conduit à la création de plusieurs start-ups, telles que Miriad Technologies, spécialisée dans l'analyse des données massives, et Fieldscale, qui propose des solutions de simulation électromagnétique basées sur l'intelligence artificielle.

1.1.3. Présentation économique et juridique

L'INRIA est l'un des 6 EPSTs, établissement public à caractère scientifique et technologique. L'INRIA est à la fois sous la tutelle du ministère de l'éducation nationale, de la recherche et de la technologie et sous la tutelle du ministère de l'économie, des finances et de l'industrie.

Depuis 2020, l'INRIA publie à la place du bilan social annuel, un Rapport Social Unique.

Les informations marquantes qu'on peut tirer du RSU de 2020 (2) :

- Effectifs globaux :
 - 42% des agents à l'INRIA sont rémunérés par d'autres organismes partenaires collaborateurs dont 2/3 sont issus d'universités et écoles françaises ;
 - 4851 agents participent aux missions d'INRIA ;
 - 28% sont des ressortissants étrangers.
- Effectifs rémunérés par l'INRIA :
 - 1/3 sont des femmes ;
 - la moyenne d'âge est de 36 ans ;
 - il y a 600 doctorants dont 269 recrutés cette année, un record à l'INRIA.
- Concours chercheurs :
 - maintenant l'INRA recrute des chercheurs sur contrat à durée indéterminée associé à un service d'enseignement dans un établissement supérieur partenaire d'INRIA ;
 - il y a eu 48 recrutements cette année.
- Le handicap :

- l'INRIA a travaillé cette année sur un projet de convention avec le FIPHFP (Fonds pour l'Insertion des Personnes Handicapées dans la Fonction Publique) ;
- le taux d'emploi d'agents en situation de handicap était de 2,82% cette année, et devrait continuer sa hausse
- La parité :
 - les femmes représentent 22% des scientifiques et 11% des responsables d'équipe de recherche à l'INRIA ;
 - à travers son plan Egalité Professionnelle femmes/hommes, l'INRIA s'est fixé l'objectif d'au moins 30% de femmes parmi les responsables d'équipes de recherche.

L'INRIA a également un impact économique important en France. En effet, les équipes-projets de l'institut travaillent en collaboration avec des entreprises, notamment des start-ups, ce qui permet de transformer leurs résultats de recherche en applications industrielles innovantes. Les partenariats avec l'industrie sont une priorité pour l'INRIA, qui souhaite valoriser ses résultats de recherche pour soutenir l'innovation technologique et stimuler la croissance économique en France. En 2020, l'INRIA a déposé 120 brevets et signé 267 accords de licence, témoignant de l'importance de la recherche et de l'innovation pour l'institut. L'INRIA contribue également au développement de l'écosystème de l'innovation en France, en encourageant la création d'entreprises innovantes par ses chercheurs et ingénieurs, ainsi qu'en organisant des événements et en soutenant des initiatives entrepreneuriales.

1.1.4. Production

A travers le COP (Contrat d'Objectifs de Performance) de 2019-2023 entre l'Etat et l'INRIA (3), l'ambition stratégique de l'INRIA est d'accélérer la construction d'un leadership scientifique, technologique et industriel, dans et par le numérique, de la France dans une dynamique européenne, en s'engageant pleinement dans le développement d'universités de recherche de rang mondial, au cœur d'écosystèmes entrepreneuriaux et industriels dynamisés par le numérique.

Dans cette perspective, l'INRIA a développé de nombreux projets de recherche dans divers domaines, tels que l'intelligence artificielle, les réseaux et systèmes distribués, la cybersécurité, les sciences des données, la modélisation et la simulation, la robotique et bien d'autres encore. Ces projets de recherche ont conduit à de nombreuses avancées scientifiques et technologiques, qui ont été valorisées à travers des brevets, des publications scientifiques, des logiciels et des collaborations avec des partenaires industriels. L'INRIA a également créé plusieurs startups et a participé à la création d'écosystèmes entrepreneuriaux pour favoriser le transfert de technologie et l'innovation dans le domaine du numérique.

1.2. Equipe

1.2.1. Introduction

Le travail de l'équipe de recherche avec laquelle j'ai collaboré porte sur l'étude des propriétés temporelles des composants cyber des Systèmes Cyber-Physiques (SCP), qui sont formés de composants cyber/informatiques et physiques communiquant entre eux. Ce sujet d'étude se concentre notamment sur l'estimation des temps d'exécution et l'ordonnancement des tâches, et se décline en plusieurs sujets de recherche.

L'équipe aborde plusieurs sujets de recherches :

- Proposition d'une classification des facteurs de variabilité des temps d'exécution d'un programme par rapport aux fonctionnalités du processeur.
- Définition d'une règle de composition de modèles statistiques basée sur des approches bayésiennes pour des bornes sur les temps d'exécution des programmes.
- Conception d'algorithmes d'ordonnancement prenant en compte l'interaction entre différents facteurs de variabilité.
- Démonstration des analyses d'ordonnancement pour les algorithmes d'ordonnancement proposés.
- Planification des programmes communiquant via des réseaux prévisibles et non prévisibles.

Un tel domaine d'études peut permettre à l'équipe de travailler en collaboration avec des partenaires académiques et industriels en France et à l'étranger dans les secteurs de l'avionique, de l'automobile et du ferroviaire. Elle peut aussi participer également à des projets de recherche européens et internationaux visant à développer des outils pour la conception et la vérification de systèmes cyber-physiques. (4)

1.2.2. Travaux

Les propriétés temporelles (temps d'exécution d'un programme ou ordonnançabilité d'un ensemble de programmes communiquant) des composants cyber des SCP sont étudiées.

Un SCP (Système Cyber-Physique) est formé de :

- Composants cyber/informatiques ;
- Composants physiques (qui communiquent entre eux).

Les études de l'équipe sont généralement faites sur des voitures, avions et drones. Un composant cyber est généralement formé de fonctions possédant des niveaux différents de criticité relativement aux propriétés temporelles.

Une solution est appropriée à un niveau de criticité si toutes les fonctions correspondantes remplissent les exigences de ce niveau.

En fonction de leurs fondements mathématiques, les solutions sont :

- Soit non probabilistes quand toutes les propriétés temporelles sont estimées ou bornées par des valeurs numériques
- Soit probabilistes quand au moins une propriété temporelle est estimée ou bornée par une fonction de distribution.

L'équipe peut se fixer trois objectifs :

- L'estimation des temps d'exécution pire cas d'un programme ;
- La décision d'un ordonnancement de tous les programmes s'exécutant dans le même composant cyber, compte tenu d'un budget énergétique ;
- La décision de l'ordonnancement de tous les programmes communiquant par des réseaux prédictibles et non prédictibles, compte tenu d'un budget énergétique

Le programme de recherche de l'équipe peut alors s'organiser autour de trois grands axes de recherche afin de répondre à ces trois objectifs :

- L'estimation des temps d'exécution pire cas ;
- La construction des repères basés sur des mesures ;
- L'ordonnancement des tâches sur différentes ressources.

2. Problématique

2.1. Contours

2.1.1. Préambule

Les travaux de cette année suivent les travaux de la première année. Ces précédents travaux avaient pour objectif la prédiction de temps d'exécution des programmes d'un autopilote de drone (*PX4* (5)) grâce à ses données environnementales. Un drone avec un autopilote est un drone qui se pilote lui-même en fonction de son environnement à partir d'un point de départ et d'un point d'arrivée qu'on lui a donné. Il capte périodiquement des données de son environnement. Etant donné l'imperfection que représente l'environnement, son trajet n'est pas parfait et il doit continuellement s'adapter. Autrement dit, son microcontrôleur doit activer des programmes liés aux composantes de mouvement de manière périodique pour répondre aux contraintes environnementales. Afin de ne pas avoir à faire fonctionner un véritable, on dispose d'un jumeau numérique et d'un logiciel (*Gazebo* (6)) permettant de simuler des données environnementales. On peut alors simuler de manière périodique des données environnementales à l'autopilote dont le microcontrôleur y réagit en exécutant des programmes.

A la fin d'une telle simulation de vol, on peut récupérer d'un côté les données environnementales simulées horodatées et d'un autre côté les données d'exécution de programmes horodatées aussi. Le travail de l'année précédente a alors été de proposer un alignement des données, et des prédictions des temps d'exécutions par programme selon les données environnementales. Le premier objectif était d'ordre exploratoire et a finalement permis une base pour les travaux de la deuxième année.

2.1.2. Contexte

L'équipe travaille sur les systèmes embarqués temps-réel, qui sont des systèmes spécifiques à un appareil et ont des contraintes de temps très strictes. L'axe majeur du travail sur les systèmes embarqués temps-réel est la prédétermination du comportement de tels systèmes. Dans la prédétermination du comportement de systèmes embarqués, cela peut inclure la prédiction des temps d'exécution des tâches du système par exemple. Les tâches du système sont elles déterminées par une politique d'ordonnancement temps-réel.

L'un des axes de travail de l'équipe concerne alors l'ordonnancement temps-réel, qui diffère de l'ordonnancement non temps-réel notamment par ses objectifs et ses contraintes. En effet, l'ordonnancement temps-réel vise à garantir que le système répondra de manière fiable et prévisible aux exigences de temps, alors que l'ordonnancement non temps-réel a pour objectif principal de maximiser l'utilisation des ressources et d'optimiser la performance globale du

système. De plus, l'ordonnancement temps-réel doit gérer des contraintes de temps très strictes, tandis que l'ordonnancement non temps-réel n'a pas à gérer de contraintes de temps aussi strictes. Les travaux de recherche d'un doctorant membre de l'équipe que j'assiste, se concentrent sur des politiques d'ordonnancement temps-réel prenant en compte des temps d'exécution de tâches probabilistes.

Un membre de l'équipe a utilisé un outil simulateur d'ordonnancement temps-réel, *SimSo* (7). Son code *Python* est disponible en open-source. Il permet de simuler des ordonnancements temps-réel qui prennent en entrée des paramètres tels que la politique d'ordonnancement, le type de processeur et un ensemble de tâches choisis par l'utilisateur. *SimSo* permet également de visualiser toutes les métriques induites par les paramètres proposés par l'utilisateur, notamment avec un diagramme de Gantt. On a alors souhaité proposer une version probabiliste de *SimSo*, c'est-à-dire qui pourrait prendre en compte des temps d'exécution probabilistes, afin d'évaluer des politiques d'ordonnancement temps-réel prenant en compte ces temps d'exécution probabilistes.

Le travail de ce projet consiste à proposer une solution pour intégrer la possibilité pour l'utilisateur de choisir ses temps d'exécution probabilistes à *SimSo* à travers son interface. L'utilisateur, ce qui permettrait d'élargir le champ des possibilités en termes d'évaluation de politiques d'ordonnancement temps-réel. Le projet s'est déroulé au sein de l'INRIA. Il consistait en une aide ponctuelle à un membre de l'équipe. Il n'est pas lié aux objectifs listés de l'équipe et est donc d'ordre open-source.

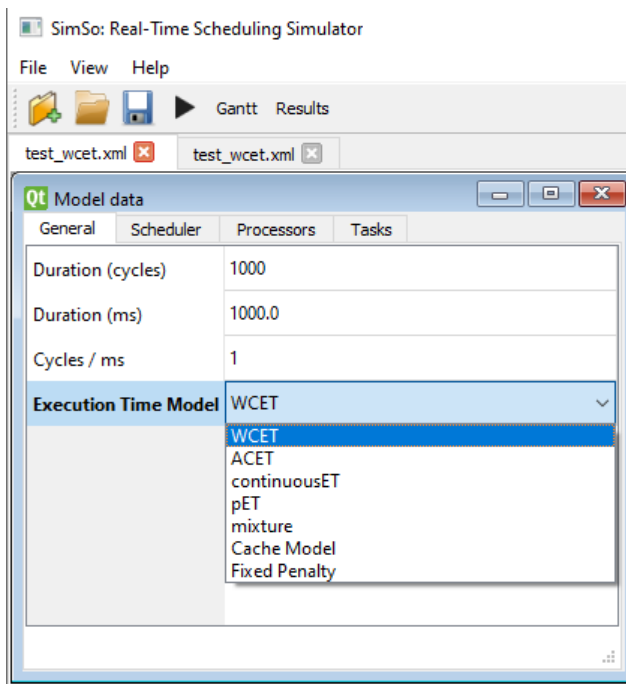


Figure 1 Capture d'écran de l'interface utilisateur de *SimSo* (paramètres d'entrée de base)

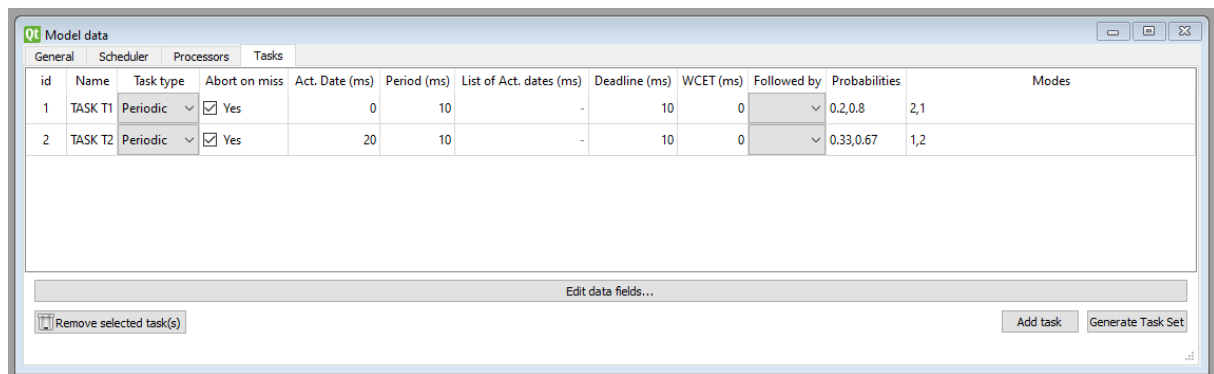


Figure 2 Capture d'écran de l'interface utilisateur de SimSo (paramètres des tâches)

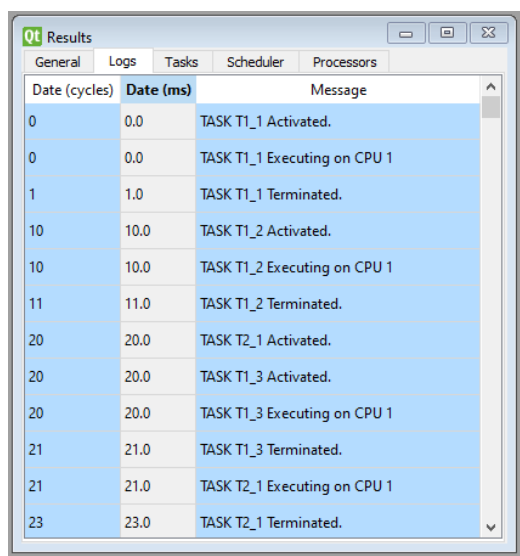


Figure 3 Capture d'écran d'une partie des métriques induites

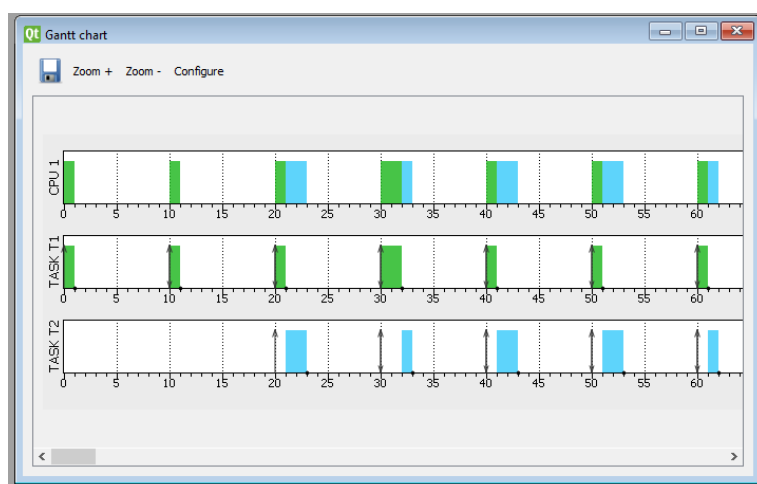


Figure 4 Capture d'écran du diagramme de Gantt induit

2.1.3. Problème rencontré

A l'heure actuelle, il existe plusieurs outils qui permettent de visualiser les métriques et le diagramme de Gantt induits par une politique d'ordonnancement temps-réel et ses différents paramètres dont un qui s'appelle *SimSo*. Mais il ne permet la possibilité de prendre en compte des temps d'exécution probabilistes.

La problématique est alors la suivante : Comment intégrer des temps d'exécution probabilistes dans l'interface utilisateur d'un outil d'ordonnancement temps-réel ?

Cette problématique peut être en conséquence divisée en plusieurs sous-problématiques :

- la visualisation des résultats de temps de réponses probabilistes grâce à l'outil *SimSo* ;
- l'ajout de la possibilité d'extraire dans des fichiers les résultats de ces politiques d'ordonnancement en veillant à ce qu'il y ait les nouvelles possibilités (les temps d'exécution probabilistes) ;
- l'assistance à un doctorant dans le travail sur le code de *SimSo*.

2.1.4. Attentes

L'objectif est donc d'améliorer l'outil *SimSo* en intégrant des temps d'exécution probabilistes dans l'interface utilisateur, afin de permettre l'ordonnancement temps-réel prenant en compte de tels types de paramètres.

Les livrables attendus à la sortie du projet sont l'outil *SimSo* amélioré, son code *Python* mis à jour pour inclure la possibilité d'ajouter des temps d'exécution probabilistes, ainsi qu'un rapport sur le projet.

2.2. Planification générale du projet

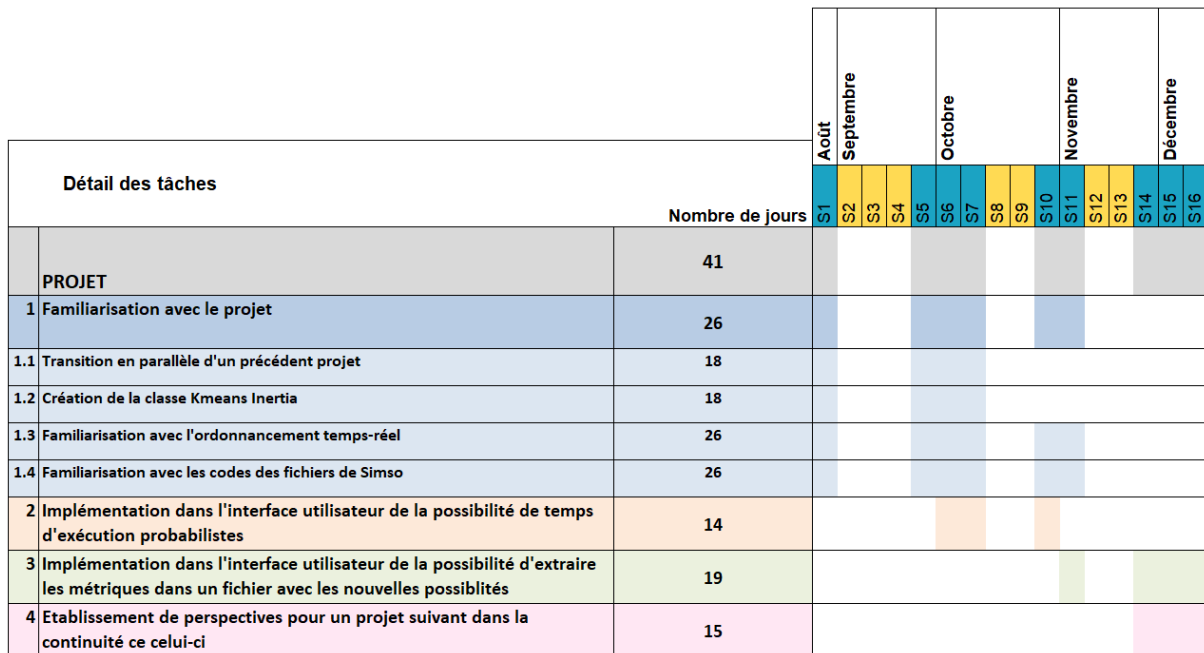


Figure 5 Diagramme de Gantt du projet

Le projet a commencé le 31 août 2022 en parallèle de la fin du projet précédent qui concernait la suite du projet de première année. Il a duré 41 jours en présentiel à l'INRIA jusqu'au 16 décembre 2022. La partie familiarisation (1.3, 1.4) est étalée sur la quasi-totalité du projet car en plus d'un début où il a fallu comprendre tout un nouvel ensemble de fichiers de codes, la familiarisation avec cet ensemble a continué durant tout le projet à laquelle se sont accompagnés des tâches (2, 3). Une fois la familiarisation confirmée, des objectifs et des perspectives (4) pour la suite du projet avaient pu commencer à être fixés, dont ont notamment résulté une fiche projet. Le projet s'est arrêté le 16 décembre.

3. Méthodologie

3.1. Choix de solution

Il existe plusieurs outils simulateurs d'ordonnancement temps-réel codé en *Python* disponible en open-source dont *SimSo*. Celui-ci est aussi disponible sur navigateur, codé en *Javascript*. Les précédents travaux de programmation de prédiction de temps d'exécution de tâches de systèmes embarqués temps-réel du doctorant que j'assistais ayant été faits sous *Python*, par défaut c'est la version application *Python* qui a été privilégié à la version navigateur *Javascript*. Le travail sous *Python* qu'il m'a proposé s'inscrivait *de facto* directement dans le sien. Cela aurait pris beaucoup de temps à adapter le code *Python* existant en *JavaScript*. Bien que la version navigateur *JavaScript* aurait pu permettre une plus grande accessibilité pour la

communauté temps-réel, ça n'était pas notre objectif initial, donc nous sommes restés sous *Python*.

3.2. Tests de validation

Pour valider le fonctionnement de l'outil, il a fallu, en tant qu'utilisateur, premièrement proposer des paramètres d'entrée simples et vérifier que les résultats correspondaient avec des résultats qu'on pourrait obtenir « à la main ». Ensuite, il a fallu vérifier que l'outil fonctionnait avec des entrées extrêmes ou qu'il pouvait les gérer. Enfin, il a fallu tester l'outil avec des paramètres d'entrée aberrants. Par exemple, si une probabilité de temps d'exécution proposée par l'utilisateur est une chaîne de caractères il faut bien que l'outil signale le problème et empêche l'exécution. Aussi, par exemple, si l'utilisateur veut proposer 2 temps d'exécution possibles dont le premier avec une probabilité d'advenir de 0.4 et le deuxième de 0.60000001, il faut que l'outil signale que la somme des probabilités est supérieure à 1, etc.

3.3. Tâches et actions

En termes d'opportunités, ce projet a offert la possibilité de :

- publier des travaux en open source ;
- participer au travail de recherche d'un doctorant prochainement publié et présenté lors d'une conférence scientifique.

Le doctorant Kévin Zagalo a été directement sollicité pour le projet.

Le budget de mon projet est de 41 jours homme.

L'essentiel du projet constitue de la programmation *Python*, l'outil *SimSo* étant codé en *Python*. L'éditeur *Python* utilisé choisi est *Pycharm*. Il n'a pas été tenté de comparer avec d'autres éditeurs (ce qui aurait pris du temps) car c'est connu et cela a été avéré lors du projet qu'il bénéficie de nombreux avantages : son interface est assez intuitive, il fait des suggestions automatiques qui peuvent éviter des bugs ou conseiller des syntaxes plus conventionnellement correctes, les différents packages sont mis à jour et restent compatibles avec celui-ci, etc.

Le projet a été réalisé avec un collègue doctorant de l'équipe qui m'a confié la tâche de travail. Pour travailler en parallèle sur le code source du projet, nous avons utilisé des commandes *git*, afin notamment de gérer les modifications apportées au code, et à travers des dépôts sur la plateforme *GitHub*. Cette approche a permis une gestion efficace des modifications apportées au code et a facilité la collaboration entre nous.

Les activités de ce projet ont d'abord été réalisé au sein de l'équipe mais seront surtout proposés publiquement en open-source.

Un ordinateur *Microsoft Windows 10 Entreprise* avec un processeur i7 m'a été mis à disposition par l'INRIA. Je bénéficie également de nombreux logiciels gratuitement (*Word*, *Pycharm*).

3.4. Résultats obtenus

3.4.1. Utilisation de l'algorithme K-means pour l'analyse des temps d'exécution des tâches d'un système embarqué temps-réel

Une classe d'objet nommée *Kmeans Inertia* a été codée lors des travaux et est disponible sur un dépôt *GitHub* (8) et en annexe.

L'objectif du code est d'utiliser l'algorithme *K-means* pour regrouper des temps d'exécution de tâches d'un système embarqué temps-réel en différents clusters. Pour déterminer le nombre optimal de clusters, la classe *KmeansInertia* implémente l'algorithme *K-means* avec une méthode qui utilise la somme des carrés des distances intra-cluster, ou inertie, normalisée par la variance totale de l'ensemble des données.

L'algorithme *K-means* est un algorithme d'apprentissage non supervisé qui permet d'identifier des clusters d'individus selon la similarité de leurs données. Avec cet algorithme, les données sont réparties en un nombre prédéfini de clusters, en fonction de la distance entre les points de données. L'algorithme commence par choisir au hasard un ensemble de centres de clusters initiaux, puis assigne chaque point de données au centre de cluster le plus proche. Ensuite, les centres de cluster sont mis à jour pour être le barycentre de tous les points de données qui y sont associés. Ce processus est répété jusqu'à ce que les centres de cluster ne changent plus de manière significative ou jusqu'à ce qu'un nombre maximal d'itérations soit atteint. Le résultat final est un ensemble de centres de cluster et les points de données qui leur sont associés. L'algorithme K-means est souvent utilisé dans les domaines de l'analyse de données, de la reconnaissance de formes, de la segmentation d'image et de la compression de données.

Dans l'algorithme *K-means*, le nombre de classes est choisi de façon automatique en utilisant la méthode du coude. Pour cela, plusieurs modèles *KMeans* sont entraînés avec un nombre croissant de classes, et l'inertie est calculée pour chaque modèle. L'inertie mesure la somme des distances au carré entre les points de données et leur centre de cluster le plus proche. Ensuite, une métrique pondérée de l'inertie et du nombre de classes est calculée pour chaque modèle, et le modèle qui minimise cette métrique est choisi.

L'argument "*inertia*" contrôle le poids relatif de la somme des distances entre les points de données et les centres de cluster dans la métrique pondérée utilisée pour choisir le nombre de classes. Une valeur plus élevée de "*inertia*" conduit à un nombre plus élevé de classes.

choisi, car cela augmente le poids relatif de l'inertie dans la métrique pondérée. Cela signifie que l'on tolère davantage de points éloignés dans les clusters et que l'on cherche à diviser les données en plus de classes pour mieux capturer les variations dans les données.

La classe *KmeansInertia* a été pensée et construite en s'inspirant des conventions des algorithmes de *machine learning* du package *sklearn*, notamment en utilisant une approche similaire à celle de la classe *KMeans* de *sklearn.cluster*. La méthode du coude (9) a été utilisée pour déterminer le nombre optimal de clusters à utiliser dans l'algorithme K-means. De manière plus précise, la proposition décrite sur cette page : (10) a également été utilisée pour permettre à l'algorithme de choisir automatiquement le nombre optimal de clusters pour un ensemble de données donné.

Elle peut alors être un outil intéressant pour l'analyse des temps d'exécution des tâches d'un système embarqué temps-réel. Ce travail de création faisait écho aux travaux de l'année précédente sur la prédiction de temps d'exécution avec *machine learning* et a permis de se familiariser en parallèle avec l'ordonnancement temps-réel et avec les données de temps d'exécution de tâches de systèmes embarqués temps-réel. Elle a permis de regrouper les temps d'exécution en différents clusters afin d'identifier les patterns et les variations dans les données.

Le code de la classe disponible en open-source peut être utilisé sur d'autres données numériques d'un tableau à deux dimensions (nombre d'échantillons, nombre de variables).

3.4.2. Familiarisation avec *SimSo*

SimSo est un outil simulateur d'ordonnancement de système embarqué temps-réel disponible en open-source et codé en *Python*. Un vrai système embarqué temps-réel fonctionne avec un ordonnancement temps-réel avec des paramètres décidés. *SimSo* permet alors à l'utilisateur de simuler de tels systèmes en indiquant en entrée des paramètres décrivant l'architecture du système, tels que le nombre de processeurs, les caractéristiques des tâches (temps d'exécution, deadlines, priorités, etc.) et les caractéristiques de la mémoire cache.

Avant de lancer une simulation, on doit indiquer plusieurs paramètres en entrée catégorisés à travers quatre onglets : *General*, *Scheduler*, *Processors* et *Tasks*.

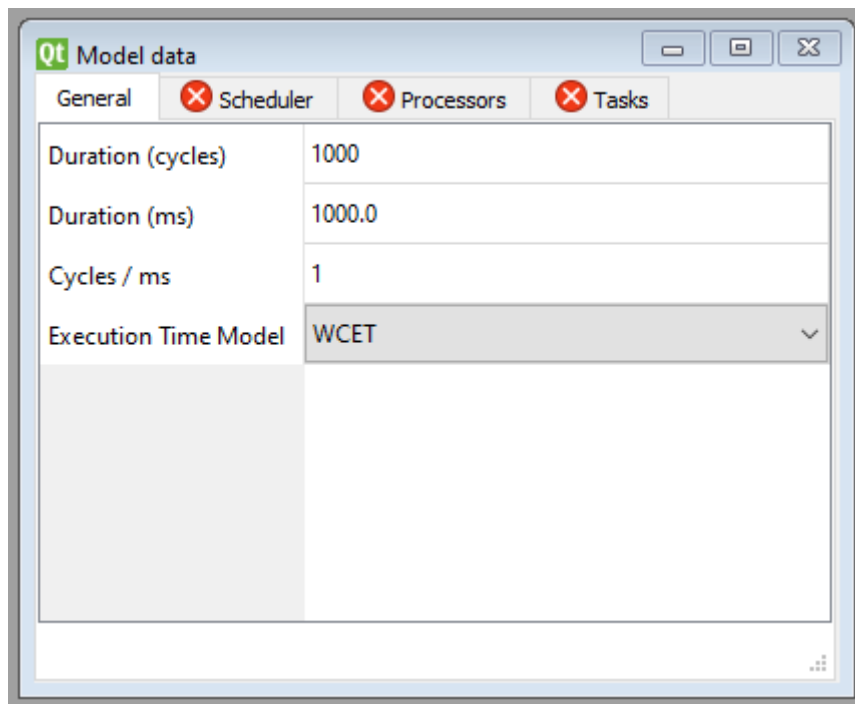


Figure 6 Capture d'écran de l'onglet General sur SimSo

Grâce à l'onglet *General*, on peut indiquer dans les paramètres d'entrée généraux qui permettent de définir les caractéristiques du système à simuler avec d'abord "Duration (cycles)", "Duration (ms)", "Cycles/ms" et "Execution Time Model".

- "Duration (cycles)" correspond à la durée totale de la simulation exprimée en nombre de cycles de processeur. Ce paramètre est important car il permet de définir la fenêtre de temps dans laquelle les tâches vont être exécutées. Il peut être choisi manuellement, sinon il se calcule automatiquement par le produit "Duration (ms)" \times "Cycles/ms".
- "Duration (ms)" correspond également à la durée totale de la simulation, mais exprimée cette fois-ci en millisecondes. Ce paramètre est utile si on veut faire correspondre la simulation à une durée réelle. Il peut être choisi manuellement, sinon il se calcule automatiquement par la division de "Duration (cycles)" par "Cycles/ms".
- "Cycles/ms" est le nombre de cycles de processeur par milliseconde. Il permet de définir la vitesse du processeur utilisé pour la simulation. Plus ce nombre est élevé, plus le processeur est rapide. Il peut être choisi manuellement, sinon il se calcule automatiquement par la division de "Duration (ms)" par "Cycles/ms".

"Execution Time Model" (souvent raccourci ETM) peut avoir une valeur possible parmi la liste : *WCET*, *ACET*, *continuousET*, *pET*, *mixture*, *Cache Model* et *Fixed Penalty*. C'est le modèle de temps d'exécution des tâches. *WCET* si on souhaite indiquer des temps d'exécution pire-cas, *ACET* si on souhaite indiquer des temps d'exécution aléatoire, *Cache Model* si on veut

que le temps de calcul soit défini par des modèles de cache statistiques et tienne compte du comportement des tâches en matière d'accès à la mémoire., *Fixed Penalty* si on veut indiquer un temps d'exécution qui sera prolongé en cas de préemption. A partir de la version de *SimSo* de base, les modèles *continuousET* et *pET* ont été ajoutés. C'est grâce à eux qu'on va pouvoir indiquer les temps d'exécution probabiliste. *continuousET* est destiné à des distributions continues de temps d'exécution et *pET* des distributions discrètes. C'est *pET* qui a été utilisé au cours de ce travail.

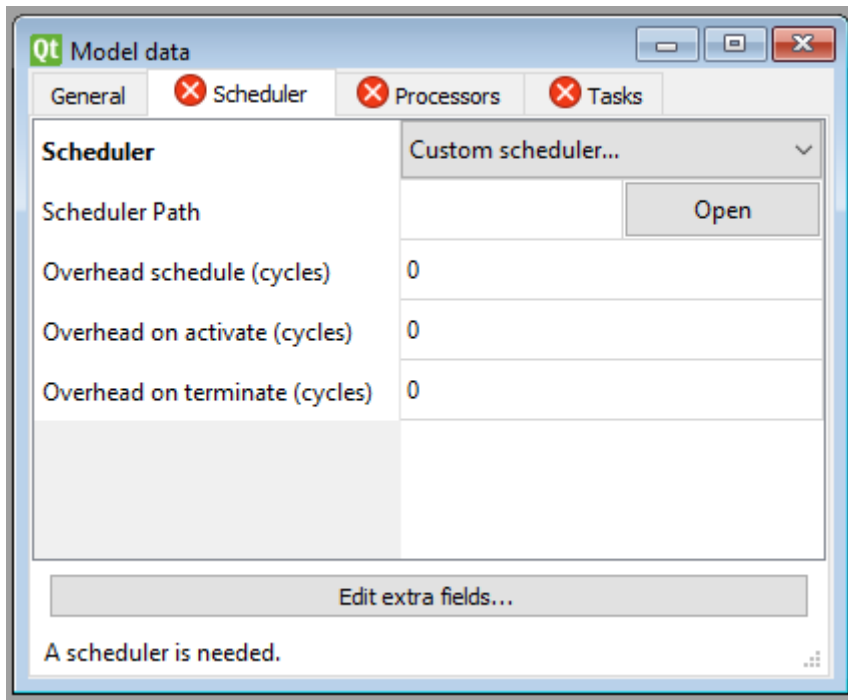


Figure 7 Capture d'écran de l'onglet Scheduler sur SimSo

Dans l'onglet *Scheduler*, on peut indiquer dans les paramètres d'entrée de l'ordonnanceur pour la simulation "Scheduler", "Scheduler Path", "Overhead schedule (cycles)" et "Overhead on activate (cycles)" et "Overhead on terminate (cycles)".

"Overhead schedule (cycles)", "Overhead on activate (cycles)", "Overhead on terminate (cycles)" sont par défaut à 0.

- L'algorithme d'ordonnancement des tâches du système peut être choisi dans une longue liste avec "Scheduler" ou en parcourant l'explorateur de fichiers avec "Scheduler Path".
- "Overhead schedule (cycles)" correspond au temps de traitement supplémentaire nécessaire pour l'exécution de l'ordonnanceur à chaque cycle de la simulation. Cette valeur est ajoutée à la durée de chaque cycle pour prendre en compte le temps nécessaire à l'ordonnanceur pour prendre une décision d'ordonnancement.

- "Overhead on activate (cycles)" correspond au temps de traitement supplémentaire nécessaire pour activer une tâche. Cela peut inclure des opérations telles que la mise à jour de la file d'attente des tâches prêtes ou l'ajout de la tâche à l'ordonnanceur.
- "Overhead on terminate (cycles)" correspond au temps de traitement supplémentaire nécessaire pour terminer une tâche. Cela peut inclure des opérations telles que la suppression de la tâche de l'ordonnanceur ou la mise à jour de la file d'attente des tâches prêtes.

L'algorithme d'ordonnancement utilisé a été seulement EDF (*Earliest Deadline First*) car il est simple et son unique utilisation permet de mieux comparer tous les résultats obtenus. La politique de l'algorithme EDF est de donner la priorité à la tâche qui a l'échéance la plus proche, c'est-à-dire celle qui doit être terminée avant toutes les autres. Ainsi, lorsque plusieurs tâches sont prêtes à être exécutées, l'ordonnanceur EDF choisit la tâche avec l'échéance la plus proche pour l'exécuter en premier.

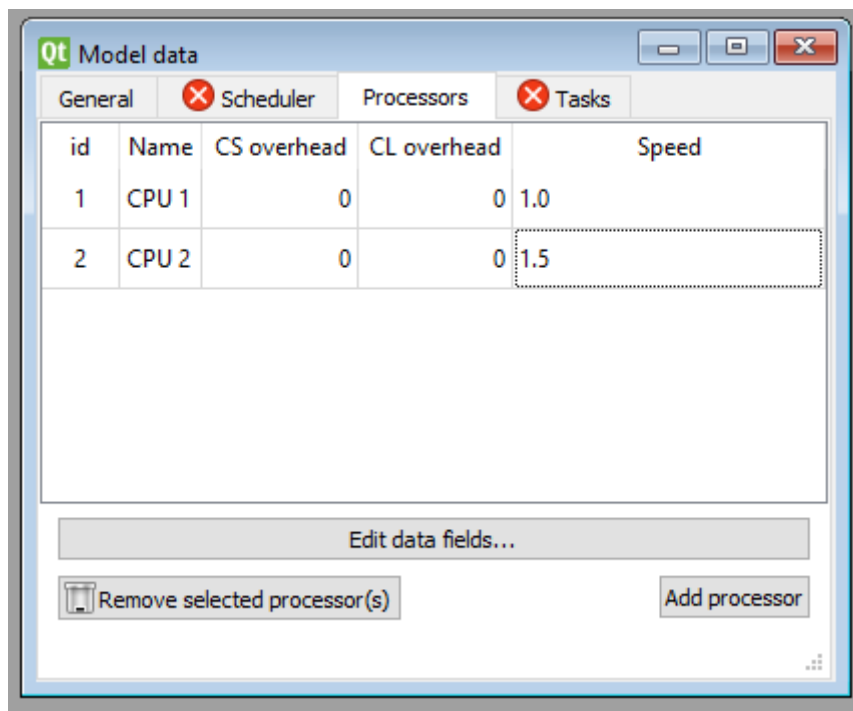


Figure 8 Capture d'écran de l'onglet Processurs sur SimSo

On peut préciser autant de processeurs au système qu'on veut simuler grâce à l'onglet "Processors".

Quand on ajoute un processeur, celui-ci a plusieurs caractéristiques : "id", "Name", "CS overhead", "CL overhead" et "Speed".

- "id" et "Name" sont des caractéristiques textuelles qui servent simplement à identifier le processeur.

- "CS overhead" correspond au temps supplémentaire nécessaire pour passer d'une tâche à une autre sur ce processeur. Ce temps supplémentaire comprend toutes les opérations nécessaires pour effectuer ce changement.
- "CL overhead" correspond au temps supplémentaire nécessaire pour démarrer ou terminer une tâche sur ce processeur.
- "Speed" correspond à la vitesse à laquelle le processeur peut traiter les instructions. Cette valeur est utilisée pour calculer le temps d'exécution de chaque tâche en fonction de son nombre de cycles et donc pour déterminer le temps total nécessaire pour traiter chaque tâche sur ce processeur.

3.4.3. Implémentation de la possibilité d'ajouter des temps d'exécution probabilistes

L'outil *SimSo* amélioré permet à l'utilisateur de simuler l'ordonnancement d'un système embarqué temps-réel prenant en compte des temps d'exécution probabilistes, lorsque le modèle de temps d'exécution choisi est *pET*.

Par défaut, il n'y a qu'une seule probabilité, de 1 donc, qui implique un temps d'exécution de 1 microseconde.

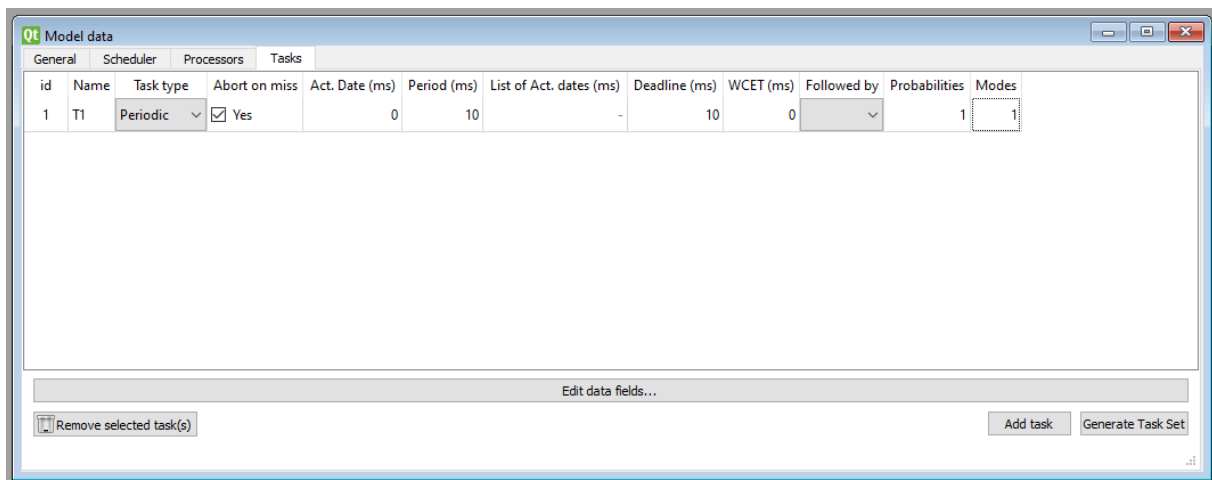
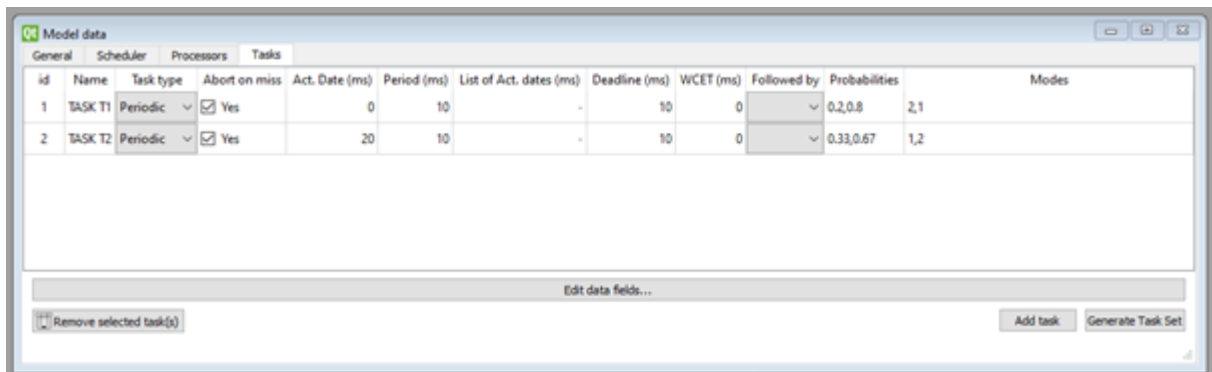


Figure 9 Capture d'écran de l'onglet Tasks avec une tâche par défaut

Si par exemple un utilisateur sélectionne les paramètres d'entrée qu'il souhaite dont la tâche 1 qu'il souhaite qu'elle ait une probabilité de 0.2 de durer 2 microsecondes et 1 sinon et dont

la tâche 2 qu'il souhaite qu'elle ait une probabilité de 0.33 de durer 1 microseconde et 2 sinon.



id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	Probabilities	Modes
1	TASK T1	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0		0.2,0.8	2,1
2	TASK T2	Periodic	<input checked="" type="checkbox"/> Yes	20	10	-	10	0		0.33,0.67	1,2

Buttons: Remove selected task(s), Add task, Generate Task Set

Figure 10 Capture d'écran de l'onglets Tasks

On voit bien à travers le diagramme de Gantt que la tâche 1 a eu un temps d'exécution de 2 microsecondes environ 20% du temps et 1 microseconde 80% du temps. Il voit aussi que la tâche 2 a eu un temps d'exécution de 1 microseconde 33% du temps et de 2 microsecondes 67% du temps.

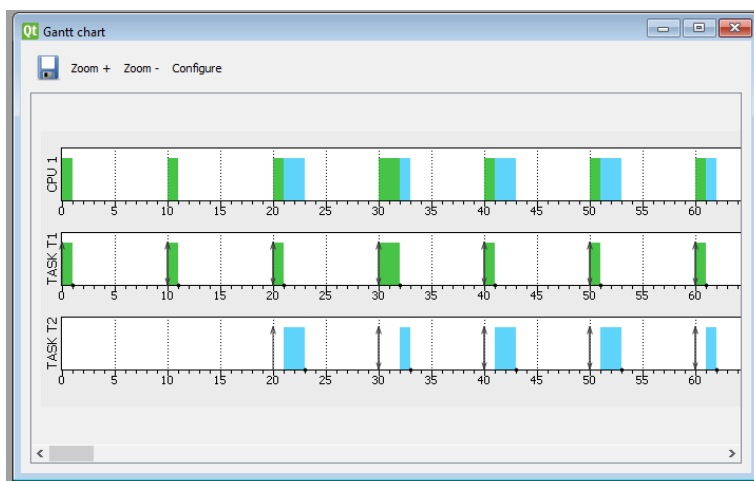


Figure 11 Capture d'écran du diagramme de Gantt

Cet ajout a alors impliqué de prévoir des possibilités d'erreurs :

- Une somme de probabilités non égale à 1 ;
- Un nombre de probabilités et un nombre de temps d'exécution non égaux.

On voit dans les captures d'écran suivantes que *SimSo* affiche une erreur car la somme des probabilités n'est pas de 1. La simulation ne peut pas alors se lancer.

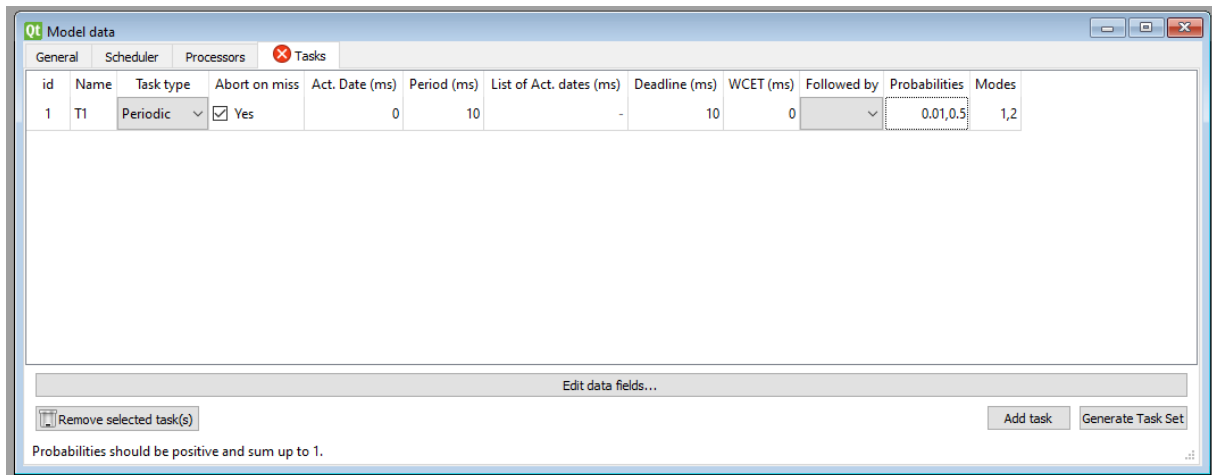


Figure 12 Capture d'écran de l'onglets Tasks avec la première erreur

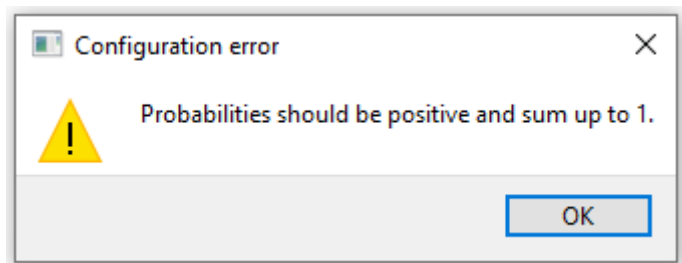


Figure 13 Capture d'écran de la fenêtre de la première erreur

On voit dans les captures d'écran suivantes que *SimSo* affiche une erreur car il n'y a pas autant de modes de temps d'exécution qu'il y a de probabilités. La simulation ne peut pas alors se lancer.

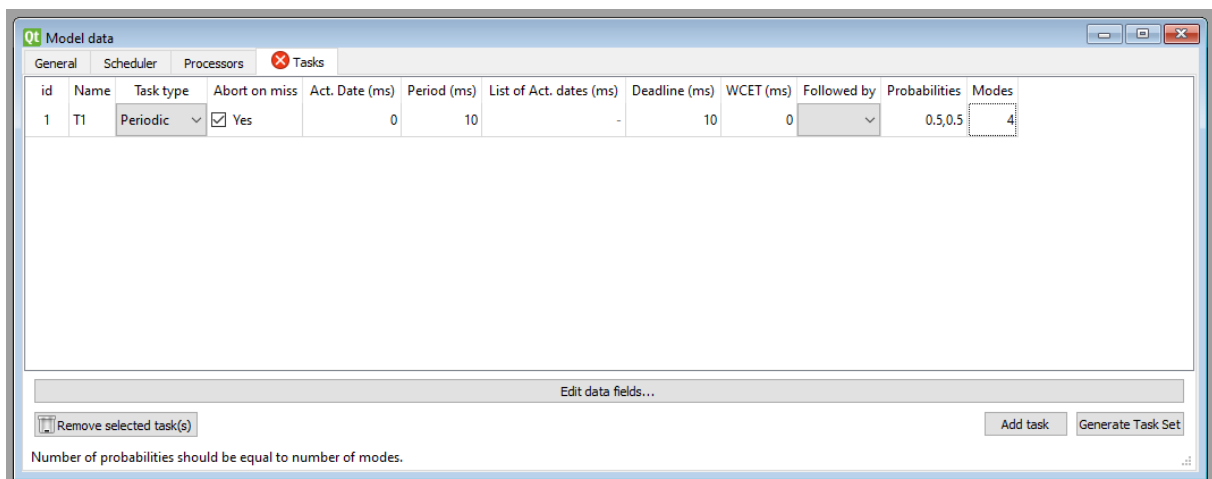


Figure 14 Capture d'écran de l'onglet Tasks avec la deuxième erreur

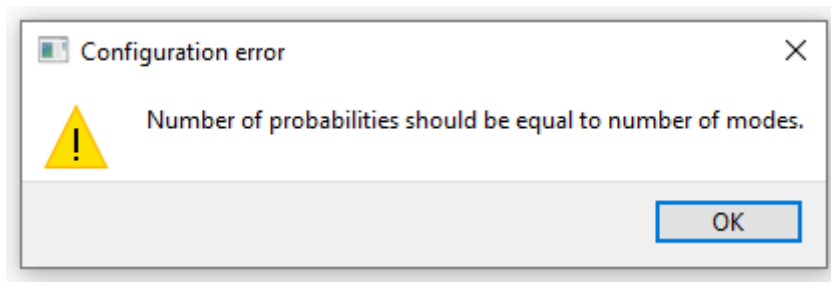


Figure 15 Capture d'écran de la fenêtre de la deuxième erreur

Quand on ajoute une tâche dans le cadre d'un modèle de temps d'exécution probabilistes, celle-ci a plusieurs caractéristiques : "id", "Name", "Task Type", "Abort on miss", "Act. Date (ms)", "Period (ms)", "List of Act. dates (ms)", "Deadline (ms)", "WCET (ms)", "Followed by", "Probabilities", et "Modes".

- "id" et "Name" sont des caractéristiques textuelles qui servent simplement à identifier la tâche.
- "Task Type" permet de préciser le type de la tâche. Par exemple, il peut s'agir d'une tâche périodique, sporadique ou apériodique. Cette information permet à l'ordonnanceur de savoir comment planifier et allouer les ressources pour chaque type de tâche.
- "Abort on miss" permet de définir le comportement de la tâche en cas de dépassement de la deadline. Si cette option est activée, la tâche sera annulée si elle ne peut pas être exécutée avant sa deadline. Sinon, la tâche peut être exécutée même si elle dépasse sa deadline.
- "Act. Date (ms)" correspond à la date de début de la première exécution de la tâche, dans le cas de tâche sporadique et apériodique.
- "Period (ms)" correspond à la période de la tâche périodique, elle permet de planifier l'exécution de la tâche de manière régulière.
- "List of Act. dates (ms)" correspond à une liste de dates d'activation pour les tâches sporadiques et apériodiques et permet de déterminer quand la tâche peut être activée.
- "Deadline (ms)" correspond à la deadline de la tâche et représente le temps limite au-delà duquel la tâche doit être terminée.
- "WCET (ms)" (*Worst Case Execution Time*) correspond au temps d'exécution le plus long que la tâche peut prendre pour être terminée. Cette information permet de planifier le temps nécessaire pour exécuter la tâche.
- "Followed by" est pour l'instant un paramètre pas encore établi (il doit permettre par la suite de pouvoir proposer une loi de probabilité que les temps d'exécution suivent).

- "Modes" et "Probabilities" correspondent aux temps d'exécutions probabilistes et à leur probabilité comme expliqué précédemment.

Au lancement de la simulation, on peut d'une part récupérer les résultats sous forme de diagramme de Gantt, mais également toutes les métriques induites par l'ordonnancement à travers la fenêtre "Results" qui a cinq onglets : *General*, *Logs*, *Tasks*, *Scheduler*, *Processors*.

Pour illustrer les exemples suivants, on va s'attarder sur la configuration de tâches suivantes :

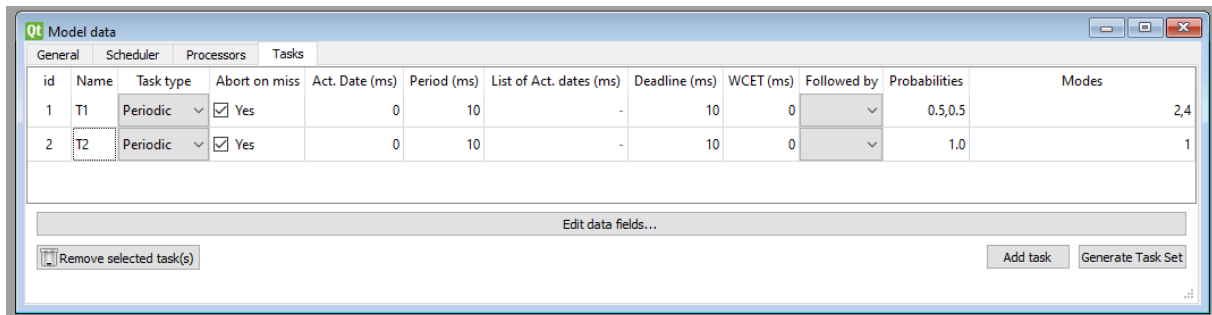


Figure 16 Capture d'écran de l'onglets Tasks

L'onglet *General* propose un tableau avec en lignes chaque CPU (et une dernière ligne les moyennes de tous les CPU) et en colonne : "Total load", "Payload" et "System load".

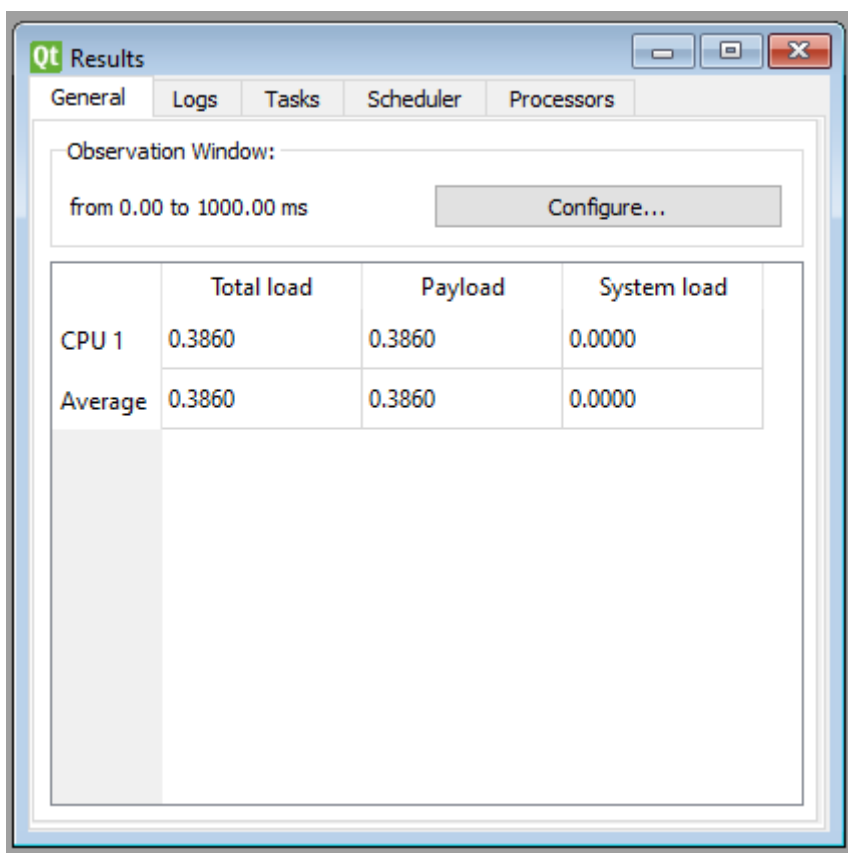
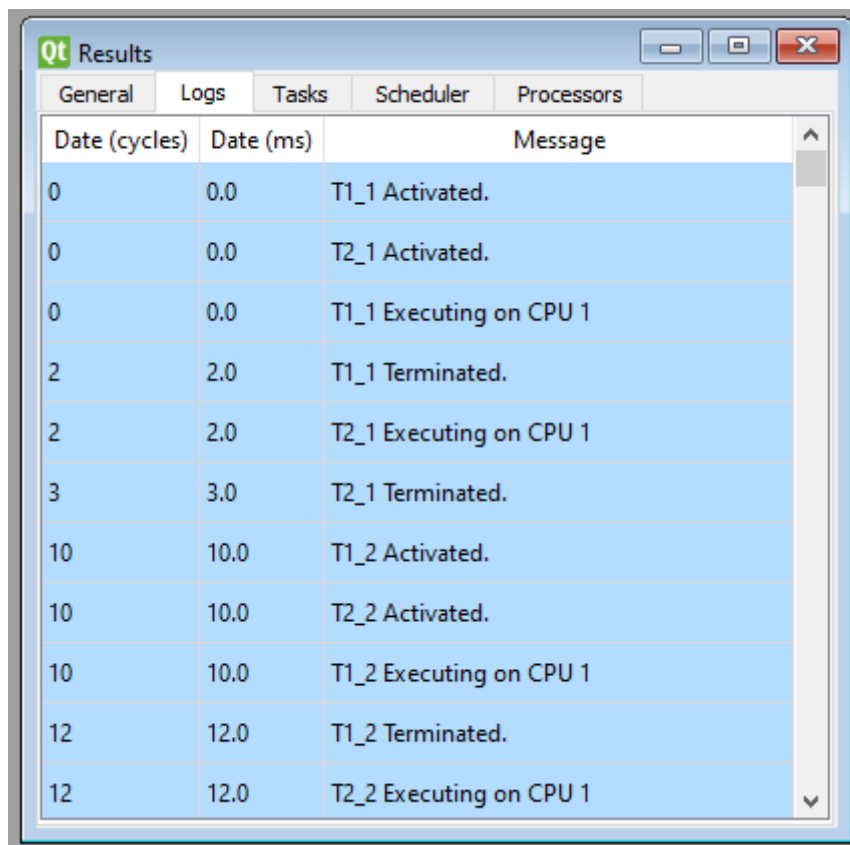


Figure 17 Capture d'écran de l'onglet General de la fenêtre Results

- La colonne "Total load" représente la charge totale du CPU, c'est-à-dire la proportion de temps d'exécution de toutes les tâches sur la durée de la simulation. Plus la charge est élevée, plus le processeur est sollicité.
- La colonne "Payload" représente la charge utile du CPU, c'est-à-dire la proportion de temps d'exécution des tâches actives (en cours d'exécution) sur la durée de la simulation. Cette colonne ne prend donc pas en compte les temps d'exécution des changements de contexte, qui ne sont pas considérés comme de la charge utile.
- La colonne "System load" représente la charge globale du système, c'est-à-dire la proportion de temps d'exécution de toutes les tâches, y compris les changements de contexte et les tâches inactives (en attente de leur prochaine activation), sur la durée de la simulation.

L'onglet *Logs* propose un tableau avec en colonne : "Date (cycles)", "Date (ms)" et "Message". Le terme "Logs" signifie un enregistrement chronologique de messages. L'onglet *Logs* contient alors un très long tableau qui affiche les différents messages liés à l'exécution des tâches.



Date (cycles)	Date (ms)	Message
0	0.0	T1_1 Activated.
0	0.0	T2_1 Activated.
0	0.0	T1_1 Executing on CPU 1
2	2.0	T1_1 Terminated.
2	2.0	T2_1 Executing on CPU 1
3	3.0	T2_1 Terminated.
10	10.0	T1_2 Activated.
10	10.0	T2_2 Activated.
10	10.0	T1_2 Executing on CPU 1
12	12.0	T1_2 Terminated.
12	12.0	T2_2 Executing on CPU 1

Figure 18 Capture d'écran de l'onglet Logs de la fenêtre Results

- La colonne "Date (cycles)" correspond au temps en cycles d'horloge de la simulation où le message a été enregistré.

- La colonne "Date (ms)" affiche la même information, mais en temps réel, en microsecondes.
- La colonne "Message" contient le contenu du message enregistré, qui peut inclure des informations sur les événements qui se sont produits dans la simulation, des avertissements ou des erreurs.

L'onglet *Tasks* propose plusieurs sous-onglets, un premier général et les autres donnant des informations pour chaque tâche.

Le sous-onglet général a plusieurs parties, on va s'attarder seulement sur la partie *Computation time*. Elle présente en colonne l'identifiant des tâches, le minimum de temps d'exécution d'une tâche, la moyenne de temps d'exécution, le maximum de temps d'exécution, l'écart-type et le pourcentage de temps pendant lequel le processeur est occupé à exécuter cette tâche, chaque ligne représentant une tâche.

Computation time:					
Task	min	avg	max	std dev	occupancy
T1	2.000	2.860	4.000	0.990	0.286
T2	1.000	1.000	1.000	0.000	0.100

Preemptions:

Migrations:

Task migrations:

Response time:

Figure 19 Capture d'écran de l'onglet *Tasks* de la fenêtre *Results*

Par exemple, le plus petit temps d'exécution qu'a eu la tâche T1 est de 2 microsecondes et son plus gros temps d'exécution a été de 4 microsecondes. En moyenne, son temps d'exécution a été de 2.860 microsecondes.

L'onglet *Scheduler* permet d'afficher des informations sur les performances et les événements liés à l'ordonnancement utilisé dans la simulation.

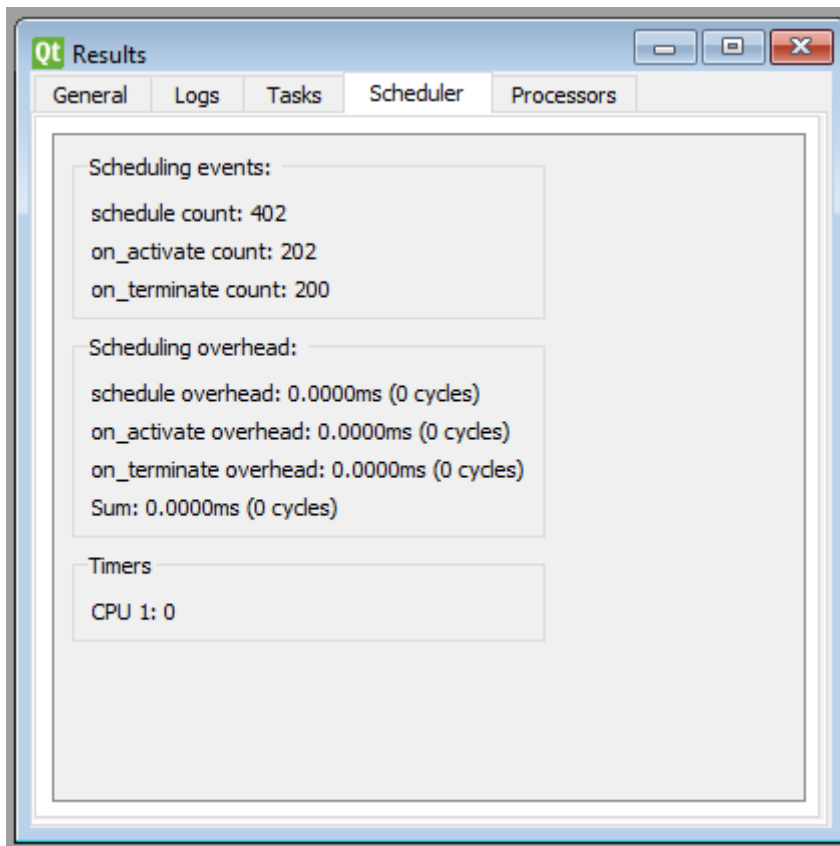


Figure 20 Capture d'écran de l'onglet *Scheduler* de la fenêtre *Results*

Cet onglet est divisé en trois parties : *Scheduling events*, *Scheduling overhead* et *Timers*.

La partie "Scheduling events" présente des statistiques sur le nombre de fois que certaines fonctions d'ordonnancement ont été appelées pendant la simulation.

- "schedule count" indique le nombre de fois que la fonction de planification principale a été appelée.
- "on_activate count" indique le nombre de fois que la fonction appelée lorsqu'une tâche est activée (c'est-à-dire qu'elle est prête à être exécutée) a été appelée.
- "on_terminate count" indique le nombre de fois que la fonction appelée lorsqu'une tâche a terminé son exécution a été appelée.

La partie "Scheduling overhead" donne des informations sur le temps de traitement nécessaire pour appeler ces fonctions d'ordonnancement.

- "schedule overhead" donne le temps de traitement moyen nécessaire pour effectuer une planification de toutes les tâches dans le système.

- "on_activate schedule overhead" et "on_terminate schedule overhead" donnent le temps de traitement moyen nécessaire pour activer ou terminer une tâche.
- "Sum" donne la somme des temps de traitement pour l'ensemble de ces fonctions.

La partie "Timers" indique le temps d'activité du processeur.

- "CPU 1" donne le temps total d'utilisation du processeur 1 pendant la simulation.

En résumé, l'onglet *Scheduler* permet de mesurer les performances de l'ordonnanceur utilisé dans la simulation en fournissant des statistiques sur le nombre de fois que les fonctions d'ordonnancement ont été appelées et sur le temps de traitement nécessaire pour effectuer ces appels. Il fournit également des informations sur le temps d'utilisation du processeur.

L'onglet *Processors* permet d'afficher les informations liées à chacun des processeurs utilisés dans la simulation.

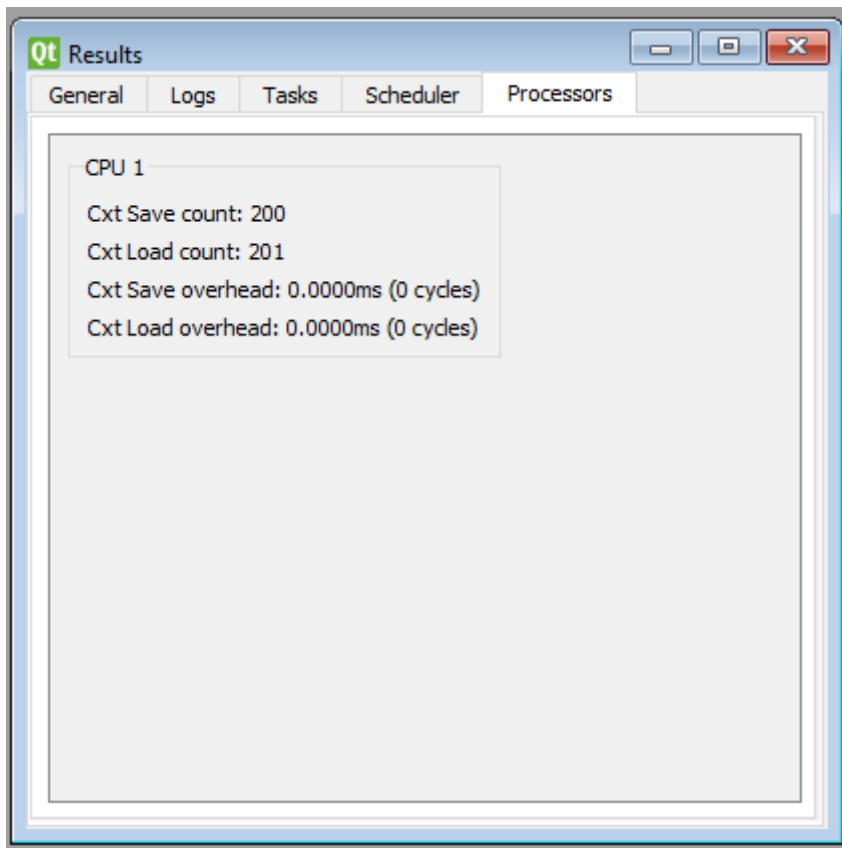


Figure 21 Capture d'écran de l'onglet Processors de la fenêtre Results

Pour chaque CPU, les valeurs présentées sont les suivantes :

- "Cxt Save count" : le nombre de changements de contexte sauvegardés (c'est-à-dire le nombre de fois où le contexte d'exécution d'une tâche a été sauvegardé avant de passer à l'exécution d'une autre tâche).

- "Cxt Load count" : le nombre de changements de contexte chargés (c'est-à-dire le nombre de fois où le contexte d'exécution d'une tâche a été chargé pour reprendre son exécution).
- "Cxt Save overhead" : le temps (en millisecondes et en cycles) pris pour sauvegarder le contexte d'exécution d'une tâche.
- "Cxt Load overhead" : le temps (en millisecondes et en cycles) pris pour charger le contexte d'exécution d'une tâche.

3.4.4. Implémentation de la possibilité d'extraire les données dans des fichiers, avec prise en compte des nouvelles fonctionnalités

L'utilisateur pouvait déjà importer ou exporter les ensembles de paramètres qu'il souhaitait utiliser ou sauvegarder dans un fichier .xml.

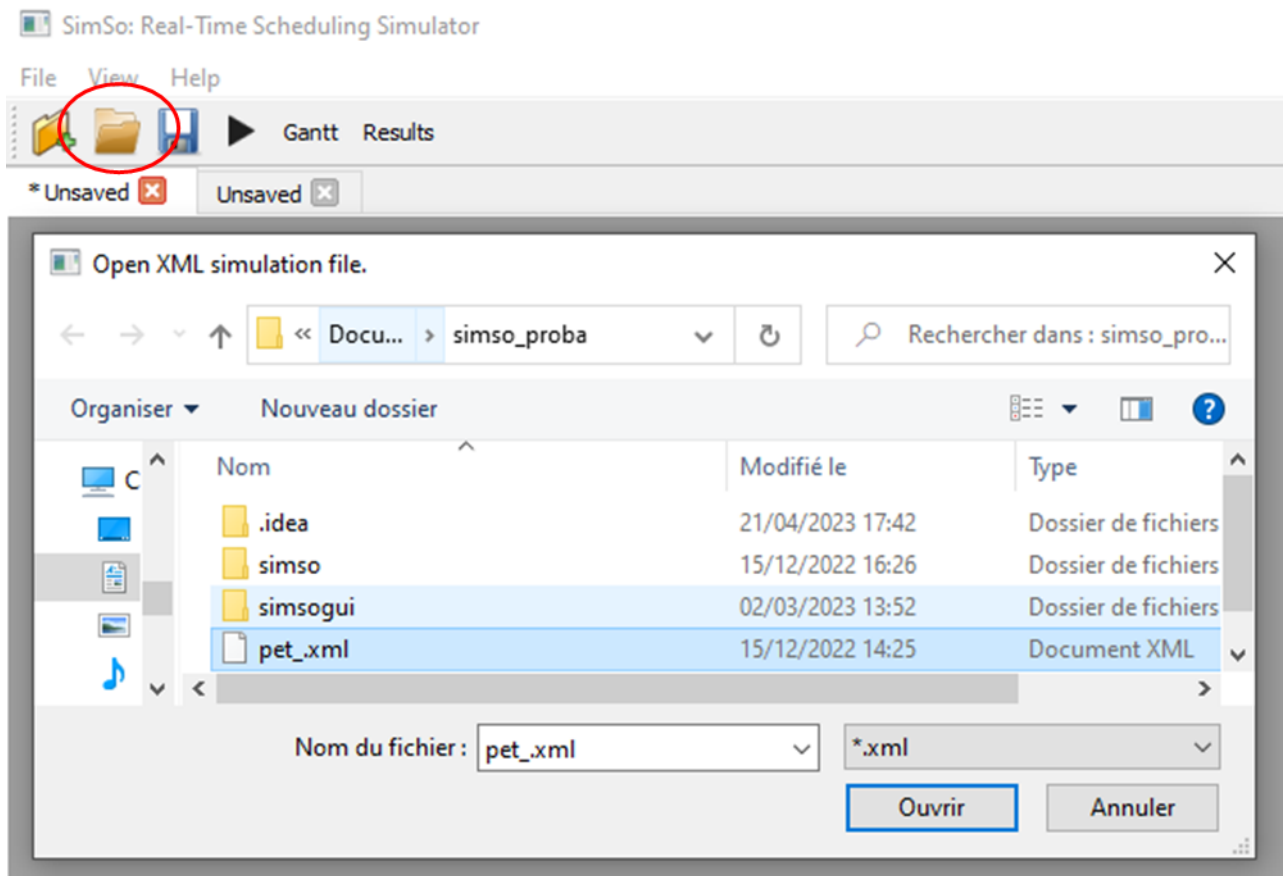


Figure 22 Capture d'écran du bouton Ouvrir

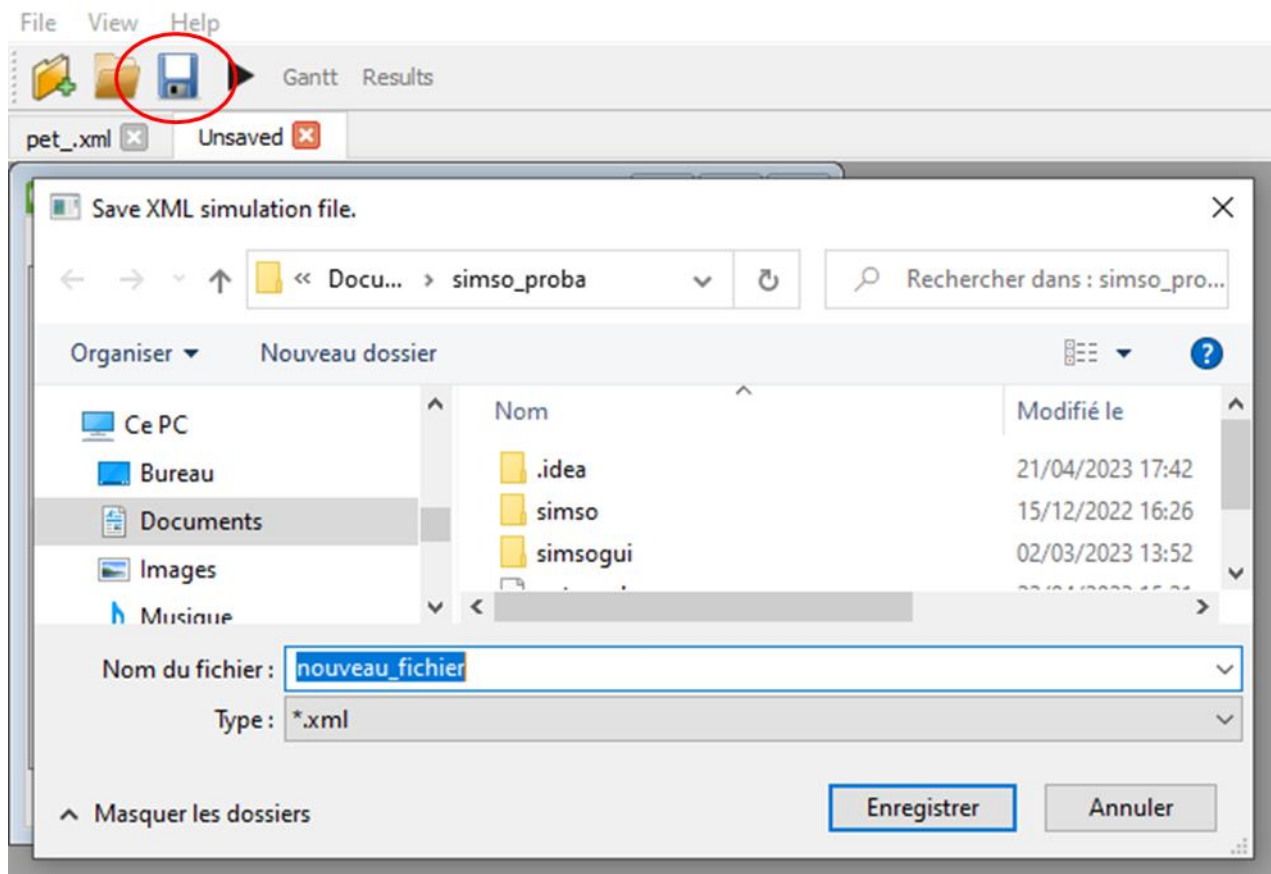


Figure 23 Capture d'écran du bouton Enregistrer

Mais depuis l'ajout des nouvelles fonctionnalités, cette fonctionnalité n'était plus effective. Il a alors fallu la mettre à jour pour ajouter les nouveaux paramètres (modes, proba) en gérant bien leur typage et il a fallu également vérifier l'exactitude des typages de tous les paramètres. Pour vérifier que ce travail était fonctionnel, sans aucune exactitude il a fallu importer, exporter, importer, exporter, etc. plusieurs fois les mêmes paramètres pour valider qu'il n'y avait aucun changement dans les données du .xml et dans les données présentes sur l'interface utilisateur.

```
<?xml version="1.0" ?>
<simulation duration="1000" cycles_per_ms="1" etm="pet">
  <sched overhead="0" overhead_activate="0" overhead_terminate="0" class="simso.schedulers.EDF"/>
  <caches memory_access_time="100"/>
  <processors>
    <processor name="CPU 1" id="1" cl_overhead="0" cs_overhead="0" speed="1.5"/>
    <processor name="CPU 2" id="2" cl_overhead="0" cs_overhead="0" speed="2.0"/>
  </processors>
  <tasks>
    <task name="TASK T1" id="1" task_type="Periodic" abort_on_miss="yes" period="20.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T2" id="2" task_type="Periodic" abort_on_miss="yes" period="32.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T3" id="3" task_type="Periodic" abort_on_miss="yes" period="44.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T4" id="4" task_type="Periodic" abort_on_miss="yes" period="56.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T5" id="5" task_type="Periodic" abort_on_miss="yes" period="68.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T6" id="6" task_type="Periodic" abort_on_miss="yes" period="80.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T7" id="7" task_type="Periodic" abort_on_miss="yes" period="92.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T8" id="8" task_type="Periodic" abort_on_miss="yes" period="104.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T9" id="9" task_type="Periodic" abort_on_miss="yes" period="116.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
    <task name="TASK T10" id="10" task_type="Periodic" abort_on_miss="yes" period="128.0" activationDate="0.0" list_activation_dates="" deadline="10.0"/>
  </tasks>
</simulation>
```

Figure 24 Capture d'écran du fichier xml (1/2)

```

0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.7,0.3" modes="3,6"/>
0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.6,0.4" modes="10,12"/>
0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.7,0.3" modes="8,13"/>
0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.9,0.1" modes="10,20"/>
0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.8,0.2" modes="40,52"/>
0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.75,0.25" modes="30,58"/>
0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.75,0.25" modes="40,50"/>
1.0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.75,0.25" modes="50,51"/>
1.0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.75,0.25" modes="30,58"/>
10.0" base_cpi="1.0" instructions="0" mix="0.5" WCET="0.0" ACET="0.0" preemption_cost="0" et_stddev="0.0" proba="0.75,0.25" modes="40,50"/>

```

Figure 25 Capture d'écran du fichier xml (2/2)

id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	Probabilities	Modes
1	TASK T1	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.7,0.3	3,6
2	TASK T2	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.6,0.4	10,12
3	TASK T3	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.7,0.3	8,13
4	TASK T4	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.9,0.1	10,20
5	TASK T5	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.8,0.2	40,52
6	TASK T6	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.75,0.25	30,58
7	TASK T7	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.75,0.25	40,50
8	TASK T8	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.75,0.25	50,51
9	TASK T9	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.75,0.25	30,58
10	TASK T10	Periodic	<input checked="" type="checkbox"/> Yes	0	10	-	10	0	<input type="checkbox"/>	0.75,0.25	40,50

Figure 26 Capture d'écran de l'onglets Tasks avec les paramètres issus du fichier xml

On voit bien que les paramètres de proba/Probabilités et modes sont les mêmes et que l'outil fonctionne avec.

3.4.5. Ouvertures

Au départ, l'idée était de réaliser un projet plus long et un ensemble de tâches plus étoffé. Cependant, ce projet n'a pas été gardé. Les travaux présentés dans ce mémoire sous forme de projet sont donc une première partie de ce projet annulé, qui était censé inclure plusieurs autres tâches. Ces différentes idées de tâches initialement prévues pour le projet avorté peuvent ainsi être considérées comme des ouvertures pour des travaux futurs.

Les différentes idées d'ouvertures :

- Implémentation de nouvelles méthodes probabilistes dans l'outil de simulation d'ordonnancement temps-réel ;
- Intégration de la programmation dynamique dans l'outil de simulation d'ordonnancement temps-réel ;
- Possibilité pour l'utilisateur de sélectionner des données correspondant aux données d'environnement d'exécution du système simulé ;
- Modélisation des temps d'exécution probabilistes à l'aide d'un modèle de machine learning prenant en variables explicatives ces variables environnementales ;
- Optimisation des performances du système temps-réel en fonction des variables environnementales sélectionnées.

La première idée de tâche aurait été d'implémenter dans l'outil des possibilités de nouvelles méthodes probabilistes qui auraient permis de modéliser selon les entrées de l'utilisateur, des temps d'exécution des tâches probabilistes, notamment des méthodes de *machine learning*. Par la suite, il avait été également énoncé la possibilité d'introduire de la programmation dynamique à l'outil.

La programmation dynamique est une méthode algorithmique pour résoudre des problèmes d'optimisation en divisant le problème en sous-problèmes plus petits et en stockant les résultats dans une table pour éviter de les recalculer. Cette approche permet de trouver des solutions optimales pour les problèmes d'optimisation. (11)

La programmation dynamique diffère du *machine learning* en ce sens qu'elle n'implique pas l'apprentissage à partir de données, contrairement au *machine learning*, qui utilise des modèles statistiques et des algorithmes pour apprendre à partir de données et prendre des décisions en conséquence.

L'implémentation de la programmation dynamique dans l'outil de simulation d'ordonnancement temps-réel pourrait permettre de trouver des solutions optimales pour les problèmes d'ordonnancement. Cependant, cela nécessiterait une réflexion plus approfondie sur la manière dont la programmation dynamique peut être adaptée à l'outil, ainsi que sur les avantages et les inconvénients de cette approche par rapport à d'autres méthodes d'optimisation.

Ensuite, une idée de tâche qui aurait permis de faire un lien très concret avec les travaux de première année était de proposer à l'utilisateur la possibilité de sélectionner des données correspondant aux données d'environnement d'exécution du système simulé. Ces données d'environnement simulées auraient été émises en même temps que l'ordonnancement est fait

et auraient été visualisables pour l'utilisateur. Elles auraient ainsi pu aussi être utilisées dans les méthodes de *machine learning* permettant de modéliser des temps d'exécution probabilistes.

Pour aller plus en détails sur l'idée de permettre à l'utilisateur de sélectionner plusieurs types de données environnementales décrivant l'environnement du système temps-réel dont il souhaite simuler l'ordonnancement, ces données environnementales pourraient être émises en même temps que l'ordonnancement des tâches et seraient visibles pour l'utilisateur. Par exemple, dans la figure 8, l'utilisateur pourrait sélectionner différentes données telles que la température, la pression, le taux d'humidité, etc. La figure 2 montre ensuite comment ces données pourraient être intégrées à la simulation, avec une ligne temporelle représentant l'évolution des données de température au fil du temps.

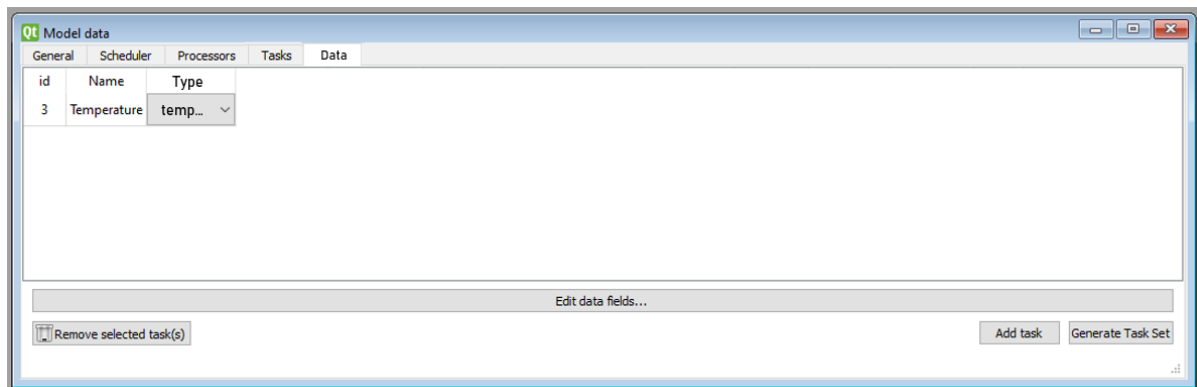


Figure 27 Montage de SimSo avec nouveaux paramètres

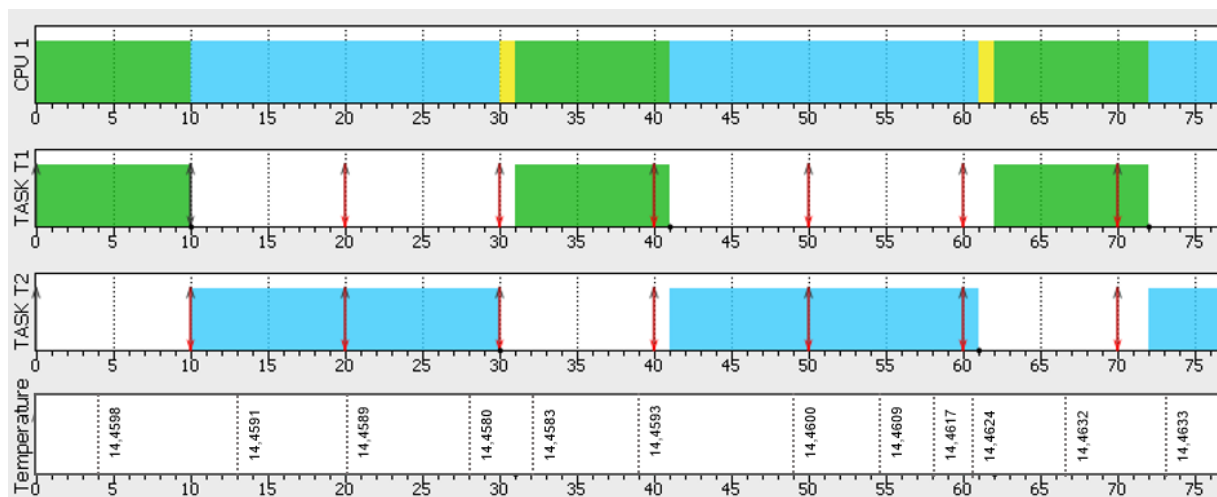


Figure 28 Montage de diagramme de Gantt avec températures du système

L'ajout de la possibilité pour l'utilisateur de sélectionner des variables de données environnementales concernant l'environnement dans lequel le système est exécuté dans l'outil de simulation d'ordonnancement temps-réel, ainsi que la possibilité de modéliser les temps

d'exécution de ses tâches à l'aide d'un modèle de *machine learning* prenant en variables explicatives ces variables environnementales, pourrait permettre à l'utilisateur de définir le système temps-réel qu'il souhaite simuler en choisissant les variables d'environnement et leur impact sur les temps d'exécution des tâches. Cette fonctionnalité lui permettrait de simuler un système qui se rapproche de son environnement réel, afin de s'adapter à des problèmes potentiels et d'ajuster les exécutions des tâches du système en conséquence.

La possibilité de modéliser les temps d'exécution des tâches à l'aide d'un modèle de *machine learning* pourrait même être utile pour optimiser le système temps-réel. En fonction des variables environnementales sélectionnées, l'utilisateur pourrait prédire les temps d'exécution des tâches avec le modèle. Cela permettrait à l'utilisateur de mettre en évidence l'influence de l'environnement sur les temps d'exécution des tâches, et de déterminer les paramètres qui peuvent influencer ces temps d'exécution. L'utilisateur pourrait alors ajuster le système en fonction de l'environnement dans lequel il fonctionne, en optimisant les paramètres pour améliorer les performances du système.

Les modélisations que l'utilisateur choisirait pourraient venir d'une simple curiosité de tester, en particulier de comparer les résultats de différents tests selon différentes modélisations. Mais un intérêt plus intuitif serait que l'utilisateur aimerait appliquer des modèles qu'il a lui-même calculé soit de son côté, soit à partir de données qu'il aurait issues depuis *SimSo*.

Cela donnerait en plusieurs étapes :

- Première étape : lancement d'une simulation sur *SimSo* avec précision de certaines variables de données de l'environnement d'exécution du système à simuler et avec une tâche avec un temps d'exécution fixe.
- Deuxième étape : récupération des données de temps d'exécution et des données environnementales.
- Troisième étape : modélisation avec le type de modèle de *machine learning* souhaité, les données précédentes de variables d'environnement en tant que variables explicatives et les données précédentes de temps d'exécution de la tâche.
- Quatrième étape : lancement d'une simulation sur *SimSo* avec précision des mêmes types de variables environnementales et avec une tâche dont le temps d'exécution est modélisé par le modèle précédent.

L'utilisateur pourrait par la suite regarder les résultats et procéder à des évaluations de ceux-ci. Par ailleurs, pour une raison scientifique, il pourrait répéter l'opération sur les nouvelles données préalablement récupérées et boucler pour essayer d'observer une potentielle convergence.

Etant donné que les travaux qui suivraient ces ouvertures seraient réalisés sur cet outil de simulation de systèmes embarqués temps-réel plutôt que sur de tels vrais systèmes, il s'agirait davantage de travaux de recherches recherche que de travaux industriels

Finalement, la possibilité de sélectionner des variables d'environnement et de modéliser les temps d'exécution des tâches à l'aide d'un modèle de *machine learning* dans l'outil de simulation d'ordonnancement temps-réel offrirait permettrait des possibilités de travaux pour les personnes travaillant sur des systèmes temps-réel et le *machine learning*. Par conséquent, cette fonctionnalité pourrait aider à optimiser le système en fonction de l'environnement réel et à améliorer les performances globales du système.

Enfin, la dernière idée de tâche était de faire tout ce qui avait été déjà fait pendant ce projet sur la version navigateur *Javascript*. (12)

Conclusion

En conclusion, ce mémoire aborde la problématique de la planification de tâches dans les systèmes embarqués temps-réel. Dans un premier temps, les contours de la problématique ont été étudiés en détaillant le contexte et les attentes du projet. Ensuite, la méthodologie utilisée s'est concentrée sur le choix de solutions, les tests de validation, les tâches et actions à effectuer, ainsi que l'analyse des résultats obtenus.

Le travail a été axé sur l'utilisation de l'algorithme *K-means* pour l'analyse des temps d'exécution des tâches et l'implémentation de temps d'exécution probabilistes. De plus, la possibilité d'extraire les données dans des fichiers avec prise en compte des nouvelles fonctionnalités de la mémoire a été implémentée grâce à la familiarisation avec *SimSo*, un simulateur de système temps-réel.

Les tests de validation ont montré que l'approche était efficace. Les travaux pourraient être approfondis en étudiant d'autres algorithmes et en prenant en compte d'autres variables telles que des variables concernant l'environnement d'exécution du système.

Bibliographie

1. inria.fr. [En ligne] <https://www.inria.fr/fr>.
2. Direction-des-ressources-humaines. Livret-synthese-RSU-2020. *inria*. [En ligne] 2020. <https://www.inria.fr/sites/default/files/2022-03/Livret-synthese-RSU-2020.pdf>.
3. Ministère de l'économie et des finances, Ministère de l'enseignement supérieur, de la recherche et de l'innovation. Contrat d'objectifs et de performance 2019-2023 entre l'Etat et l'INRIA. [En ligne] 2019. https://www.inria.fr/sites/default/files/2020-02/COP_INRIA_2019-2023_version-finalessssignatures.pdf.
4. Zhao, Wei. Scheduling Tasks with Resource Requirements in Hard Real-Time Systems. [En ligne] <https://ieeexplore.ieee.org/document/1702256>.
5. Docs PX4. [En ligne] https://docs.px4.io/main/en/flight_controller/pixhawk4.html.
6. Gazebo. [En ligne] <https://staging.gazebosim.org/home>.
7. SimSo. [En ligne] <https://projects.laas.fr/simso/>.
8. Auvray, Marc-Antoine. KmeansIntertia. [En ligne] <https://github.com/MarcAntoineAuvray/KmeansInertia/blob/main/KmeansInertia.py>.
9. Méthode du coude pour une valeur optimale de k dans KMeans. [En ligne] <https://stacklima.com/methode-du-coude-pour-une-valeur-optimale-de-k-en-kmeans/>.
10. Herman-Saffar, Or. An Approach for Choosing Number of Clusters for K-Means. [En ligne] <https://towardsdatascience.com/an-approach-for-choosing-number-of-clusters-for-k-means-c28e614ecb2c>.
11. Programmation dynamique. [En ligne] https://fr.wikipedia.org/wiki/Programmation_dynamique.
12. SimSo Configuration. [En ligne] <https://projects.laas.fr/simso/simso-web/#/configuration>.
13. Kopernic. KOPERNIC 2021 activity report. [En ligne] 2021. <https://raweb.inria.fr/rapportsactivite/RA2021/kopernic/uid0.html>.

Liste des figures

Figure 1 Capture d'écran de l'interface utilisateur de SimSo (paramètres d'entrée de base)	15
Figure 2 Capture d'écran de l'interface utilisateur de SimSo (paramètres des tâches).....	16
Figure 3 Capture d'écran d'une partie des métriques induites	16
Figure 4 Capture d'écran du diagramme de Gantt induit.....	16
Figure 5 Diagramme de Gantt du projet.....	18
Figure 6 Capture d'écran de l'onglet General sur SimSo	22
Figure 7 Capture d'écran de l'onglet Scheduler sur SimSo	23
Figure 8 Capture d'écran de l'onglet Processurs sur SimSo	24
Figure 9 Capture d'écran de l'onglet Tasks avec une tâche par défaut.....	25
Figure 10 Capture d'écran de l'onglets Tasks	26
Figure 11 Capture d'écran du diagramme de Gantt	26
Figure 12 Capture d'écran de l'onglets Tasks avec la première erreur	27
Figure 13 Capture d'écran de la fenêtre de la première erreur.....	27
Figure 14 Capture d'écran de l'onglet Tasks avec la deuxième erreur	27
Figure 15 Capture d'écran de la fenêtre de la deuxième erreur	28
Figure 16 Capture d'écran de l'onglets Tasks	29
Figure 17 Capture d'écran de l'onglet General de la fenêtre Results	29
Figure 18 Capture d'écran de l'onglet Logs de la fenêtre Results	30
Figure 19 Capture d'écran de l'onglet Tasks de la fenêtre Results	31
Figure 20 Capture d'écran de l'onglet Scheduler de la fenêtre Results	32
Figure 21 Capture d'écran de l'onglet Processors de la fenêtre Results	33
Figure 22 Capture d'écran du bouton Ouvrir.....	34
Figure 23 Capture d'écran du bouton Enregistrer	35
Figure 24 Capture d'écran du fichier xml (1/2)	35
Figure 25 Capture d'écran du fichier xml (2/2)	36
Figure 26 Capture d'écran de l'onglets Tasks avec les paramètres issus du fichier xml	36
Figure 27 Montage de SimSo avec nouveaux paramètres	38
Figure 28 Montage de diagramme de Gantt avec températures du système	38

Annexes

```
from numpy import array,
argmin from sklearn.cluster
import KMeans

# https://towardsdatascience.com/an-approach-for-choosing-number-of-clusters-for-k-means-
c28e614ecb2c class KmeansInertia:

    def __init__(self, inertia=0.1,
                 n_clusters_max=5):self.inertia_ =
                 inertia
                 self.model = None
                 self.n_clusters_ =
                 n_clusters_max

    def fit(self, X, y=None,
            sample_weight=None):print(X)
            XX = array([x for x in X if all(x)])
            inertia_0 = ((XX -
            XX.mean(axis=0))**2).sum()model_list
            = []
            inertia_list = []
            for k in range(1, self.n_clusters_):
                kmeans = KMeans(k).fit(XX,
                sample_weight=sample_weight)
                model_list.append(kmeans)
                inertia_list.append(kmeans.inertia_ / inertia_0 + self.inertia_ * k)
                model_list.append(kmeans)
            best_k =
            argmin(inertia_list)
            self.model =
            model_list[best_k]return
            self

    def predict(self, X):
        return self.model.predict(X)

    def fit_predict(self, X):
        return self.fit(X).predict(X)

    def
        get_parameter
        s(self):
        params = {}
        params["model"] = self.model
        params["model_params"] =
        self.model.get_params()params["inertia_"]
        = self.inertia_
        params["cluster_centers_"] =
        self.model.cluster_centers_return params

    def set_parameters(self,
                       params):self.model =
                       params["model"]
                       self.model.set_params(**params["model_params"])
                       self.inertia_ = params["inertia_"]
                       self.model.cluster_centers_ =
                       params["cluster_centers_"]

if __name__ == '__main__':
    X = [[1, 2, 3, 4], [2,1,4,3],[1, 2, 3, 1], [2,1,4,1],[1, 2, 3, 4], [2,2,4,3]]
    kmeans_ = KmeansInertia(inertia=0.1)
    print(kmeans_.fit_predict(X))
    print(kmeans_.get_parameters())
```

Résumé

Le travail présenté se concentre sur l'amélioration d'un simulateur de système temps-réel pour explorer l'ordonnancement des tâches et l'analyse des temps d'exécution dans les systèmes embarqués temps-réel. Travailler sur un simulateur d'ordonnancement de systèmes embarqués temps réel permet de tester et optimiser l'algorithme d'ordonnancement avant son déploiement sur des systèmes physiques, ce qui peut réduire les coûts et les risques liés aux erreurs de conception. L'une des approches employées est basée sur l'algorithme *K-means* et une deuxième sur la mise en place de temps d'exécution probabilistes.

Les travaux commencent par une étape très importante de familiarisation avec l'outil simulateur d'ordonnancement de systèmes embarqués temps-réel nommé *SimSo* codé en *Python* ainsi que la prédiction par algorithme de *machine learning* de temps d'exécution de tâches. Les résultats obtenus montrent que l'utilisation de l'algorithme *K-means* pour l'analyse des temps d'exécution des tâches peut être efficace. Aussi, la mise en place de temps d'exécution probabilistes dans *SimSo* est désormais fonctionnelle. De plus, une fonctionnalité permettant l'extraction de données dans des fichiers avec les nouveaux ajouts a été implémentée.

Mots-clés : simulateur de système temps-réel, systèmes embarqués temps-réel, temps d'exécution probabilistes, *K-means*, ordonnancement.

The presented work focuses on improving a real-time system simulator to explore task scheduling and execution time analysis in real-time embedded systems. Working on a real-time embedded system scheduling simulator allows for testing and optimizing the scheduling algorithm before it is deployed on physical systems, which can reduce the cost and risk of design errors. One of the approaches used is based on the K-means algorithm and a second on the implementation of probabilistic execution times.

The work begins with a very important step of familiarization with the real-time embedded systems scheduling simulator tool named *SimSo* coded in *Python* as well as the prediction of task execution times by machine learning algorithm. The results obtained show that the use of the K-means algorithm for the analysis of task execution times can be effective. Also, the implementation of probabilistic execution times in *SimSo* is now functional. In addition, a feature allowing the extraction of data from files with the new additions has been implemented.

Keywords: real-time system simulator, real-time embedded systems, probabilistic execution time, K-means, scheduling.