L. David, F. Cottet, E. Grolleau. *Gigue temporelle et ordonnancement par échéance dans les applications temps réel.* IEEE Conf. Inter. Francophone d'Automatique (CIFA2000), Juillet 2000, Lille, France.

Gigue temporelle et ordonnancement par échéance dans les applications temps réel

L. David, F. Cottet, E. Grolleau
LISI-ENSMA – BP 40109 – 1, Av. Clément Ader – 86961 Futuroscope Cedex
{david | cottet | grolleau} @ensma.fr

Résumé :
L'utilisation d'un ordonnancement en ligne selon les algorithmes classiques (RM, DM et ED) ne permet pas au cours de l'exécution d'une application temps réel, de maîtriser le phénomène de gigue temporelle de certaines tâches. Cette gigue s'avère d'autant plus néfaste lorsque les tâches ont une période qui doit être strictement respectée (acquisition de données, commande d'un actionneur, etc.). Nous nous proposons donc dans ce travail de fournir une méthode permettant d'annuler la gigue temporelle pour des tâches dont les contraintes de régularité sont strictes, ceci dans le cadre des ordonnancements basés sur l'échéance (DM, ED).
Mots-Clés :
Gigue temporelle, ordonnancement en ligne, algorithmes d'ordonnancement.

I INTRODUCTION

Les systèmes informatiques qui pilotent un procédé dans un contexte temps réel gèrent classiquement un ensemble de fonctions d'acquisition, au travers de capteurs, et un ensemble de commandes, par l'intermédiaire d'actionneurs. Lors du passage du domaine continu au domaine discret intervient la période d'échantillonnage, dans le processus inverse, intervient la période dite de restitution. Ces périodes, sous certaines conditions, se doivent d'être précisément respectées, ce qui, dans un contexte d'exécutions concurrentes, n'est pas toujours le cas. Cette irrégularité d'exécution de la période d'une tâche est communément illustrée par le phénomène de gigue temporelle [1], [2].

Une méthode hors-ligne d'ordonnancement avec contrôle de la gigue dans le cas réparti a été présentée par DiNatale et Stankovic dans [2]. Une autre solution hors-ligne au problème du respect de la cadence d'échantillonnage, réside dans la recherche exhaustive des séquences valides respectant ce critère [3], [4]. Nous nous intéressons ici au contexte des algorithmes d'ordonnancements en ligne, pour lesquels nous proposons une méthode permettant de maîtriser la gigue des tâches à contraintes de régularité stricte, que nous appellerons indifféremment tâches régulières.

Nous nous proposons dans un premier temps de formaliser le calcul de la gigue temporelle pour une tâche périodique. Puis nous présentons une méthode de traitement de la gigue pour des tâches régulières de durée d'exécution unitaire, et nous généralisons enfin cette méthode aux tâches de durée quelconque.

Dans cette étude, nous nous plaçons dans l'hypothèse d'un environnement monoprocesseur sur lequel sont traitées des tâches périodiques et indépendantes. Nous utilisons le modèle de Liu & Layland [5] (cf. figure I.1), où chaque tâche est caractérisée par 4 paramètres temporels :

$r_{i,1}$: date de la première activation de la tâche,
C_i : durée d'exécution maximale,
D_i : délai critique,
P_i : période de la tâche.

Nous utilisons également les notations suivantes :
$r_{i,k}$: date de réveil de la $k^{ième}$ instance de la tâche i ($r_{i,k} = r_{i,1} + (k-1)P_i$),
$d_{i,k}$: échéance de la $k^{ième}$ instance de la tache i ($d_{i,k} = r_{i,k} + D_i$),
X^* : la nouvelle valeur d'une variable X après une modification, X peut être une date de première activation ou un délai critique.

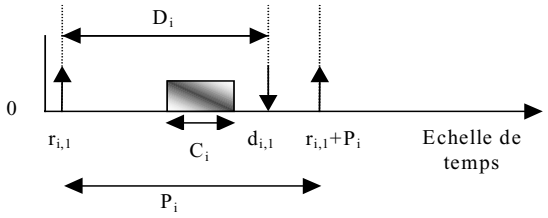

Figure I-1 *Modèle de tâches utilisé*

II LA GIGUE TEMPORELLE

Le phénomène de gigue temporelle est le terme employé en informatique temps réel pour définir l'irrégularité d'exécution d'une tâche périodique. Cette irrégularité peut être évaluée de plusieurs façons. Nous nous proposons ici de l'évaluer sur une période d'étude qui est donnée, dans le cas d'un système de tâches périodiques à départ différé, par [6], [7] :

$$P_{Etude} \leq \max_i(r_{i,1}) + 2\,ppcm(P_i) \qquad \textbf{(1)}$$

et se réduit à : $$P_{Etude} = ppcm(P_i) \qquad \textbf{(2)}$$

pour un système de tâches périodiques à départ simultané.

Formalisons maintenant ce calcul : soient $s_{i,k}$ (respectivement $e_{i,k}$) les dates de début d'exécution (resp. de fin d'exécution) de la $k^{ième}$ instance de la $i^{ième}$ tâche d'une configuration de tâches périodiques (cf. figure II.1).

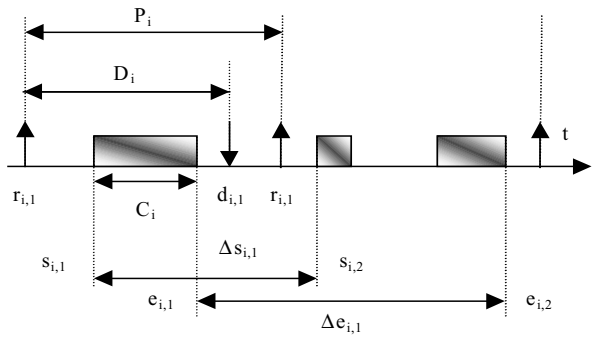

Figure II-1 *Définition des débuts et des fins d'exécution*

Les écarts entre deux débuts d'exécution (resp. entre deux fins d'exécution) concernant deux instances successives d'une tâche se définissent par :

$$\Delta s_{i,k} = s_{i,k+1} - s_{i,k} \quad (\text{resp. } \Delta e_{i,k} = e_{i,k+1} - e_{i,k})$$

où k désigne le k^{ième} écart sur la période d'étude. $\Delta s_{i,k}$ et $\Delta e_{i,k}$ varient entre les limites 0 et $2P_i$, bornes maximales dans le cas de tâches à échéance sur requête avec des durées d'exécution négligeables.

Posons ensuite : $$g_{S_{i,k}} = \frac{|\Delta s_{i,k} - P_i|}{P_i} 100\,\%$$

qui évalue, en pourcentage par rapport à P_i, l'irrégularité des débuts d'exécution de la k^{ième} instance et de la k+1^{ième} instance :

- si l'écart vaut P_i (tâche régulière), alors $g_{S_{i,k}} = 0\,\%$.
- si l'écart tend vers 0 ou vers $2P_i$ (irrégularité maximum dans l'exécution), alors $g_{S_{i,k}} = 100\,\%$.

Définition II.1 *La moyenne arithmétique de ces termes nous permet alors de définir la gigue temporelle moyenne de la tâche T_i par :*

$$G_{moy}(T_i) = \frac{1}{N_i} \sum_{k=1}^{N_i} \frac{|\Delta s_{i,k} - P_i|}{P_i} 100\,\% \qquad \textbf{(3)}$$

où N_i représente le nombre d'intervalles sur la période d'étude, entre deux instances successives. N_i s'obtient par[1] *:*

$$N_i = \left\lceil \frac{P_{Etude}}{P_i} \right\rceil - 1$$

(On suppose P_{Etude} strictement plus grande que P_i.)

Différentes études [8][9][10] ont montré les effets, parfois désastreux, de la gigue sur la qualité de transmission de l'information, au sens traitement du signal (échantillonnage, commande, etc…). Par ailleurs, les algorithmes d'ordonnancement RM, DM et ED reposent sur le modèle de Liu et Layland [5] où chaque exécution d'une tâche sur une période quelconque, est indépendante, en terme d'ordonnancement, de l'exécution de cette même tâche sur les autres périodes. Ainsi, le respect des contraintes de régularité de certaines tâches n'est pas pris en compte. Il est alors intéressant d'étudier de concert l'ordonnancement des tâches d'une configuration, ainsi que la maîtrise de la gigue temporelle de certaines tâches, tout cela, dans le contexte en ligne de RM, DM et ED.

Dans les configurations de tâches périodiques sur lesquelles nous travaillons, seules certaines sont considérées comme critiques en terme de respect strict de leur période, les autres sont périodiques sans contraintes strictes de régularité.

III METHODE DE TRAITEMENT DE LA GIGUE

Lors de la réalisation d'une application temps réel, le paramètre période P_i des tâches régulières est lié aux caractéristiques dynamiques du procédé, et ne peut donc pas être modifié. Le paramètre de durée d'exécution C_i est lui aussi supposé fixe car déterminé par le programme de la tâche. Seuls les paramètres $r_{i,1}$ et D_i peuvent, moyennant quelques précautions, être modifiés.

Le principe adopté pour diminuer la gigue, voire l'annuler, repose alors sur trois idées principales :

- 1/ décaler les $r_{i,1}$ des tâches régulières pour permettre des activations à des instants différents sur toute la période d'étude, et pour permettre à ces tâches de ne pas se concurrencer pendant leur exécution respective;
- 2/ rendre ces mêmes tâches, prioritaires par rapport aux autres non strictement régulières, pour que ces dernières ne viennent pas non plus perturber la régularité de l'exécution;
- 3/ vérifier l'ordonnançabilité de la nouvelle configuration si les délais critiques sont modifiés.

Afin de simplifier le problème, nous nous intéressons dans une première approche aux tâches régulières de durée d'exécution unitaire, la durée unitaire correspondant à la granularité de l'ordonnanceur. Cette hypothèse est ensuite levée.

III.1 Tâches régulières de durée unitaire

III.1.1 Modification des dates d'activation

Considérons 2 tâches régulières de durée unitaire : T_i et T_j. Pour ces deux tâches, nous devons éviter les perturbations à l'activation (cf. figure III.1).

Pour ces deux tâches, le décalage consiste à trouver un couple d'entiers naturels $(r_{i,1}, r_{j,1})$ vérifiant la propriété suivante:

$$(\forall k \in I\!N)(\forall k' \in I\!N)\ r_{i,1} + k \cdot P_i \neq r_{j,1} + k' \cdot P_j$$

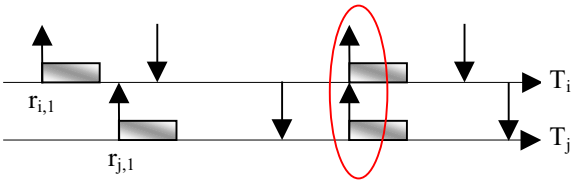

Figure III-1 - *Perturbation à l'activation de T_i par T_j*

[1] $\lceil\ \rceil$ représente la partie entière supérieure.

Ce qui, en introduisant la notation mathématique modulo (mod) et le pgcd ($\wedge$), et en se basant sur les travaux de [11] et [12], conduit à trouver un couple d'entiers naturels ($r_{i,1}$,$r_{j,1}$) vérifiant :

$$r_{j,1} - r_{i,1} \neq 0 \bmod(P_i \wedge P_j)$$

où P_i et P_j sont nécessairement non premiers entre eux.

Ce résultat doit ensuite être étendu à l'ensemble des tâches à contraintes de régularité stricte. Soit n, le cardinal de cet ensemble, le décalage des activations de ces tâches consiste alors à trouver un n-uplet ($r_{1,1}$,$r_{2,1}$,…,$r_{n,1}$) vérifiant[2] :

$$\forall (i,j) \in [\![1;n]\!]^2 \text{ avec } i \neq j,\ r_{j,1} - r_{i,1} \neq 0 \bmod(P_i \wedge P_j), \quad \textbf{(4)}$$

avec en outre une condition nécessaire sur les périodes :

Proposition III.1 *Une condition nécessaire pour la recherche des $r_{i,1}$ consiste à avoir tous les P_i non premiers entre eux deux à deux.*
(voir [12] pour détails)

De cette condition générale (4), différentes méthodes peuvent être élaborées afin de trouver un ensemble de solutions. Ainsi nous avons la proposition suivante:

Proposition III.2 *si* $\min_{i,j} [P_i \wedge P_j] \geq n$ *alors il existe des solutions au problème posé. Dans ce cas, le n-uplet $r_{1,1} = 0, \ldots, r_{n,1} = n-1$ est solution ainsi que toute permutation de ces n nombres.*
(voir [12] pour détails)

Ce problème d'arithmétique est cependant complexe, et lorsque les outils analytiques démontrent l'existence d'une solution sans pouvoir la préciser, nous faisons appel à des techniques d'énumération avec une recherche systématique de tous les $r_{i,1}$ possibles.

III.1.2 Changement de priorité

Nous devons ensuite rendre les tâches régulières prioritaires par rapport aux autres tâches de l'application. Cette action sur la priorité des tâches dépend essentiellement de l'algorithme d'ordonnancement choisi :
- en RM, la priorité étant attribuée en fonction de la période que nous considérons comme fixe, il n'est pas possible d'intervenir sur la priorité;
- en DM par contre, nous pouvons agir sur la priorité en modifiant les délais critiques. Soit T_i, une tâche à contrainte de régularité stricte. Afin d'imposer une priorité maximum à T_i par rapport aux tâches non régulières, nous choisissons un nouveau délai critique D_i^* tel que :

$$(C_i = 1) \leq D_i^* \leq \min_{j \neq i} (D_i, D_j - 1).$$

Ainsi pour j désignant l'ensemble des tâches non nécessairement régulières, T_i a un délai critique strictement inférieur à tous les D_j, et se trouve donc prioritaire;
- en ED, il suffit d'imposer un nouveau délai critique tel que $D_i^*=C_i=1$.

[2] $[\![1;n]\!]$ représente l'ensemble des entiers compris entre 1 et n (inclus).

III.1.3 Exemple d'application

Afin d'illustrer cette méthodologie, prenons l'exemple d'une configuration de 6 tâches dont deux sont des tâches d'acquisition ou de commande, de durée d'exécution unitaire, à contraintes de régularité. (cf. tableau III.1 pour commentaires)

Ces paramètres conduisent à une période d'étude de 72 unités de temps processeur, un taux d'utilisation processeur de 94,5 %, et des gigues temporelles que nous présentons dans le tableau III.2. Notons que si les tâches d'acquisition ou de commande devaient suréchantillonner le signal, la configuration ne serait pas ordonnançable.

Tableau III-1 *Exemple de configuration de tâches régulières à durée d'exécution unitaire*

Tâches	$r_{i,1}$	C_i	D_i	P_i	Commentaires
Acquisition 1	0	1	8	8	Contrôle rapide (ex. contrôle d'ouverture d'une vanne)
Traitement 1	0	2	8	8	
Contrôle 1	0	1	7	8	
Acquisition 2	0	1	18	18	Mesure lente (ex. acquisition température)
Traitement 2	0	4	17	18	
Contrôle 3	0	1	6	6	Tâche de contrôle rapide (ex. alarme)

Tableau III-2 *Calcul des gigues sur les tâches régulières*

	RM	DM	ED
Gigue Acquisition1	9,4 %	9,4 %	10,9 %
Gigue Acquisition2	18,4 %	18.5 %	11.1 %

Nous appliquons alors la méthode présentée sur les trois algorithmes d'ordonnancement. Cette application est partielle dans le cas RM où nous n'intervenons pas dans la gestion de la priorité, contrairement aux deux autres cas DM et ED. La recherche des $r_{i,1}$ utilise la proposition III.2. Ainsi, en comparant les résultats obtenus à ceux présentés, le tableau III.3 illustre la limitation de la méthode dans le cas RM et la complète maîtrise de la gigue temporelle pour DM et ED. Nous pouvons remarquer que, dans le cas RM, le simple fait de décaler les dates d'activation modifie la gigue, mais permet aussi dans certains cas de la diminuer, voire de l'annuler. Les cas DM et ED fournissent, quant à eux, les résultats prévus : une gigue nulle.

Le nombre important de possibilités concernant le décalage pourrait permettre en outre de prendre en considération d'autres critères : le nombre de changements de contexte, le temps de réponse, etc…

Tableau III-3 - *Résultat du traitement de maîtrise de la gigue des tâches d'acquisition*

	RM		DM	ED
	($r_{1,1}$=6, $r_{2,1}$=3)	($r_{1,1}$=3, $r_{2,1}$=2)	($r_{1,1}$=0, $r_{2,1}$=1) $1 \leq D_1^* \leq 5$, et $1 \leq D_2^* \leq 5$	($r_{1,1}$=0, $r_{2,1}$=1) et D_i^*=1
Gigue Acq1	8.1 %	0 %	0 %	0 %
Gigue Acq2	12.7%	15.9 %	0 %	0 %

III.2 Tâches de durée quelconque

Intéressons nous maintenant au cas des tâches dont les durées d'exécution sont plus grandes que la granularité choisie pour l'ordonnanceur. Dans ce cas, le simple décalage des dates de première activation ne suffit pas, il faut tenir compte de la durée d'exécution comme illustré sur la figure III.2. Par ailleurs, la propriété de préemptivité des tâches régulières ne va pas dans le sens d'une diminution de la régularité d'exécution.

Pour pallier ce problème, nous considérons que chaque tâche régulière est composée par une première partie à régularité stricte, celle correspondant à l'échantillonnage ou la commande, et par une seconde partie sans contrainte de régularité. Il est ensuite possible de découper cette tâche selon le principe de découpage en forme normale (cf. [13], [14], [15]) en introduisant des contraintes de précédence artificielles, assurant un séquencement correct des deux parties de la tâche.

Dans la suite, l'ensemble des tâches régulières est un ensemble de tâches pour lesquelles nous voulons une régularité stricte d'exécution, et pour lesquelles l'exécution se fait sans préemption (cf figure III.3). La méthode employée pour traiter la gigue temporelle se présente alors en 4 points :
- 1/ décalage approprié des $r_{i,1}$, pour éviter les perturbations d'exécution entre les tâches régulières (exécution sans préemption),
- 2/ redéfinition des priorités des tâches régulières pour les rendre plus prioritaires que celles non nécessairement régulières,
- 3/ redéfinition des $r_{i,1}$ et, si besoin, des échéances, pour les tâches qui sont liées par une contrainte de précédence artificielle, et ce, selon les méthodes de Blazewick [14] et Chetto [15],
- 4/ vérification de l'ordonnançabilité de la nouvelle configuration de tâches.

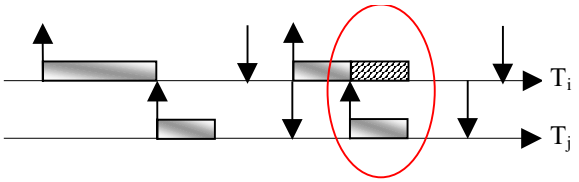

Figure III-2 *Perturbation entre deux tâches régulières dans le cas de tâches de durée quelconque : la zone hachurée de T_i rentre en conflit avec l'activation de T_j*

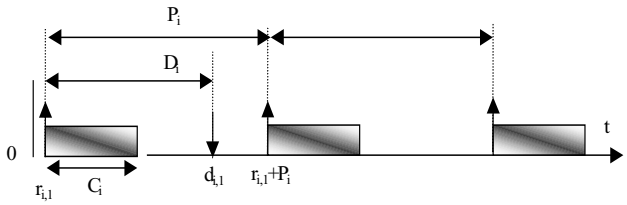

Figure III-3 *Exécution régulière et en bloc d'une tâche à contraintes de régularité stricte.*

III.2.1 Modification des dates d'activation

Dans ces conditions, le choix des dates d'activations $r_{i,1}$ des tâches régulières est plus restrictif, et il faut prendre en compte la durée d'exécution C_i des tâches régulières dans la recherche des $r_{i,1}$. La proposition décrivant les conditions nécessaires et suffisantes pour que deux tâches régulières devant s'exécuter sans préemption, ne puissent pas se concurrencer pendant leurs exécutions est décrite de façon détaillée dans [12] et utilise en fait le théorème du reste chinois généralisé [16] :

Proposition III.3 *Soient deux tâches T_i et T_j non préemptibles, s'exécutant dès leurs activations, de durée d'exécution respective C_i et C_j, et soit p leur PGCD. Ces deux tâches ne se perturbent pas entre elles pendant leur exécution si et seulement si*

$$\begin{pmatrix} r_{j,1}-r_{i,1} \neq 0 \bmod(p) \\ r_{j,1}-r_{i,1} \neq 1 \bmod(p) \\ \dots \\ r_{j,1}-r_{i,1} \neq C_i-1 \bmod(p) \end{pmatrix} et \begin{pmatrix} r_{i,1}-r_{j,1} \neq 0 \bmod(p) \\ r_{i,1}-r_{j,1} \neq 1 \bmod(p) \\ \dots \\ r_{i,1}-r_{j,1} \neq C_j-1 \bmod(p) \end{pmatrix}$$

Cette proposition peut facilement s'illustrer en disposant sur un cercle de périmètre , $p = P_i \wedge P_j$ l'ensemble des valeurs interdites pour $r_{j,1}-r_{i,1}$ (cf. figure III.4).

Avant toute recherche des $r_{i,1}$, il faut là encore vérifier une condition sur les périodes :
Proposition III.4 *Une condition nécessaire pour la recherche des $r_{i,1}$ consiste à avoir tous les P_i non premiers entre eux deux à deux.*
(voir [12] pour détails)

Comme dans le cas précédent, il faut généraliser ce résultat à toutes les tâches régulières prises deux à deux. Posons pour cela, $S_{i,j}$ le système formé des deux systèmes ci-dessus. Il s'agit alors de trouver l'ensemble des $r_{i,1}$ des tâches à contraintes de régularité stricte répondant au problème suivant :

$$\forall (i,j) \in [\![1;n]\!]^2 \text{ avec } i \neq j \text{ , } (r_{i,1} \text{ , } r_{j,1}) \text{ solution de } S_{i,j}. \quad \textbf{(5)}$$

Proposition III.5 *La figure III.4 illustre une autre condition nécessaire d'existence de solutions au problème : pour tous les couples (i,j) d'entiers, avec $i \neq j$, on doit vérifier $P_i \wedge P_j > (C_j - 1) + (C_i - 1)$.*
(voir [12] pour détails)

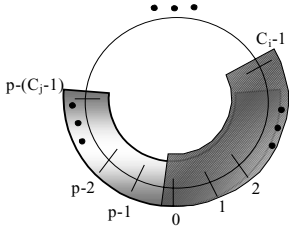

Figure III-4 *Les valeurs de $r_{j,1}$-$r_{i,1}$ interdites dans 9/p9 sont grisées avec $p = P_i \wedge P_j$*

Quelques résultats analytiques nous permettent là encore, dans certains cas de préciser des solutions [12], et dans les autres cas, nous effectuons une recherche des $r_{i,1}$ en utilisant des techniques d'énumération.

III.2.2 Le changement de priorité

Passons maintenant à la redéfinition des priorités :
- pour RM, pas de choix possible,
- pour DM, même chose que dans le cas C_i=1, à ceci près que dans l'ensemble des tâches non nécessairement régulières, nous avons aussi celles qui sont liées aux tâches régulières par les contraintes de précédence,
- pour ED, on pose $D_i^*=C_i$, ce qui rend la tâche impérativement non préemptible sans quoi, une faute temporelle se produirait et la configuration ne serait pas ordonnançable. Ici aussi, nous devons prendre en compte les tâches qui sont liées aux tâches régulières par les contraintes de précédence. En pratique cependant, la configuration de tâches obtenue peut s'avérer non ordonnançable et il faut alors relâcher les contraintes au niveau des D_i^*. Une incrémentation progressive de ce paramètre est alors réalisée en prenant en compte de nouvelles dates d'activation. En effet, pour que la faute temporelle qui traduit la non ordonnançabilité du système puisse disparaître, il suffit d'autoriser la préemption de la tâche régulière, ce qui augmente les contraintes dans la recherche des dates d'activation : il faut tenir compte de cette fin d'exécution repoussée. La gigue de la tâche est cependant bornée.

III.2.3 Prise en compte des contraintes de précédence

Il reste ensuite à traduire la relation de précédence qui existe entre certaines tâches régulières et certaines tâches non nécessairement régulières. Selon [14][15], il suffit de définir de nouvelles dates d'activation et de nouvelles priorités. L'algorithme d'ordonnancement choisi conditionne les différentes méthodes. Prenons l'indice j pour les tâches non nécessairement régulières et l'indice i pour les tâches régulières. Pour toutes les tâches non nécessairement régulières ayant une contrainte artificielle de précédence, nous définissons :
- en DM, une nouvelle date d'activation

$$r_{j,1}^{*} = \max(r_{i,1}^{*}, r_{j,1}) \qquad \textbf{(6)}$$

où $r_{i,1}^{*}$ est la nouvelle date d'activation de la tâche régulière T_i, partie non préemptible, issue de la résolution de l'équation **(5)**. Quant à la priorité, nous n'avons pas besoin d'y revenir, puisque les nouvelles priorités affectées aux tâches régulières sont, par construction, plus grandes que celles non nécessairement régulières.
- en ED, de la même façon, il suffit d'une part de prendre :

$$r_{j,1}^{*} = \max(r_{i,1}^{*} + C_i, r_{j,1}) \qquad \textbf{(7)}$$

et, d'autre part, de définir une nouvelle échéance tel que :

$$(d_{i,1}^{*})^{*} = (D_i^{*} + r_{i,1}^{*})^{*} = \min(D_i^{*} + r_{i,1}^{*}, d_{j,1} - C_j) \qquad \textbf{(8)}$$

où D_i^{*} et $r_{i,1}^{*}$ sont respectivement le nouveau délai critique et la date de première activation de la tâche T_i, issus des premières étapes de la méthode. Notons que cette redéfinition de l'échéance contraint le paramètre de la même façon que dans le paragraphe sur la priorité : il ne s'agit pas d'augmenter le paramètre d'un côté, pour le diminuer de l'autre.

III.2.4 Exemple d'application

Appliquons maintenant la méthode à un exemple de configuration de tâches (cf. tableau III.4). Deux sont à contraintes de régularité stricte, et nous décidons de rendre non préemptibles leurs deux premières unités de temps processeur. Ces paramètres conduisent à une période d'étude de 224 unités processeurs, un taux d'utilisation processeur de 89%, et nous obtenons les gigues temporelles du tableau III.5.

L'application de la méthode sur ce système de tâches a nécessité l'emploi d'une technique d'énumération de solutions concernant le décalage, et nous a conduit à scinder les deux tâches d'acquisition, pour donner finalement la configuration du tableau III.6 pour laquelle la période d'étude est de 451, et pour laquelle il existe deux contraintes artificielles de précédence.

Tableau III-4 *Exemple de configurations de tâches*

Tâches	$r_{i,1}$	C_i	D_i	P_i
Acquisition 1	0	3	16	16
Traitement 1	0	4	14	14
Contrôle 1	0	1	13	14
Acquisition 2	0	3	32	32
Traitement 2	0	4	31	32
Contrôle 3	0	1	8	8

Tableau III-5 *Gigue des tâches à contraintes de régularité pour le cas des tâches de durée quelconque*

	DM	ED
Gigue Acq1	9,6 %	9,6 %
Gigue Acq2	12,5%	12 %

La redéfinition des priorités en DM ne pose pas de problème et il en est de même pour ED dans cet exemple. Pour les deux tâches régulières, Acquisition 1.1 et Acquisition 2.1, on définit :
- en DM, (D_1^{*}, D_2^{*})=(7 ,7),
- en ED, (D_1^{*}, D_2^{*})=(2 ,2).

Définition qui permet d'annuler la gigue des deux tâches d'acquisition dans les conditions citées.

Tableau III-6 *Configuration modifiée après décalage*

Tâches	$r_{i,1}$	C_i	D_i	P_i
Acquisition 1.1	**0**	**2**	**16**	**16**
Acquisition 1.2	**0 (DM) / 2 (ED)**	**1**	**16**	**16**
Traitement 1	**0**	**4**	**14**	**14**
Contrôle 1	**0**	**1**	**13**	**14**
Acquisition 2.1	**3**	**2**	**32**	**32**
Acquisition 2.2	**3 (DM) / 5 (ED)**	**1**	**32**	**32**
Traitement 2	**0**	**4**	**31**	**32**
Contrôle 3	**0**	**1**	**8**	**8**

Dans cette partie, pour les besoins de la méthode, nous avons défini des contraintes artificielles de précédence entre des tâches régulières s'exécutant en bloc et des tâches non nécessairement régulières. Cette démarche est donc généralisable à une configuration de tâches périodiques comportant des relations de précédence, à la seule condition que les tâches régulières ne soient précédées d'aucune tâche, de sorte que les conditions de Blazewick et de Chetto sur les $r_{i,1}$ ne viennent pas s'opposer à la redéfinition des $r_{i,1}$ par notre méthode.

IV CONCLUSION

Dans le cadre des applications temps réel comprenant certaines tâches à contrainte de régularité d'exécution stricte, nous proposons une méthode permettant de conserver les ordonnancements temps réel par échéance (DM, ED). En effet, ces algorithmes d'ordonnancement, qui facilitent la validation temporelle d'une application (soit par des vérifications analytiques, soit par des simulations) ne permettent pas de contrôler la périodicité stricte d'exécution des tâches. Afin d'aller dans ce sens, la méthode décrite se décompose en plusieurs phases : décalage des dates d'activations au travers des $r_{i,1}$, redéfinition des priorités au travers des D_i, et enfin, vérification de l'ordonnançabilité. Dans le cas des tâches indépendantes à durée d'exécution unitaire (granularité de l'ordonnanceur), nous obtenons alors des tâches avec une gigue temporelle nulle. Lorsqu'il s'agit de tâches de durée d'exécution non unitaire, les résultats sont plus restrictifs, seul l'algorithme DM permet d'appliquer la méthode et d'obtenir de façon sûre des gigues nulles. Dans le cas ED, un relâchement progressif des contraintes liées aux priorités doit souvent être mis en place pour obtenir la périodicité stricte d'exécution des tâches.

Les configurations de tâches dépendantes avec ou sans partage de ressources, ainsi que le cas distribué constitueront la suite logique de ce travail. Enfin dans une prochaine étape, plutôt que d'annuler la gigue totalement, nous essaierons de spécifier une gigue maximum.

BIBLIOGRAPHIE

[1] Giorgio C. Butazzo, « Hard real-time computing systems », Kluwer Academic Publishers, p. 77-108, 1997.

[2] M. DiNatale, J.A. Stankovic, « Applicability of simulated annealing methods to real-time scheduling and jitter control » Proc. of the IEEE real-time systems symposium, Pisa, Italy, p. 190-199, Dec 5-7 1995.

[3] S.-T. Cheng, C.-M. Chen – « A cyclic scheduling approach for relative timing requirements » - Proc. of the 3rd IEEE Real-Time Applications Workshop, p. 160-163, 1996.

[4] A. Choquet-Geniet, D. Geniet, F. Cottet – « Exhaustive computation of the scheduled task execution sequences of a real-time application, Proc. of the 4th International symposium on formal techniques in real-time and fault-tolerant systems, Uppsala, Sweden, sept 1996.

[5] C.L. Liu, J.W.Layland « Scheduling algorithms for multiprogramming in a hard real time environment » - Journal of the Association for Computing Machinery (ACM), vol 20, n°1, p. 46-61 (1973).

[6] J.Y.-T. Leung, M.L. Merrill, « A note on preemptive scheduling of periodic real-time tasks », Information Processing Letters, 11(3), pp. 115-118, Nov, 1980.

[7] E. Grolleau, A. Choquet-Geniet, "Cyclicité des ordonnancements de systèmes de tâches périodiques différées", Real-Time Systems 2000, pp. 215-229, ed. Teknea, Paris, 28-30 mars 2000.

[8] A.V. Balakrishnan – « On the problem of time jitter in sampling » I.R.E. Trans. on Inf. Theory, vol. IT8, p. 226-236, 1962.

[9] A.J. Jery « The Shannon sampling theorem: its various extensions and applications, a tutorial review », Proc. of the IEEE, vol. 65, n°11, p. 1565-1596, 1977.

[10] F. Castanié « An approximately unbiased recovery of discret Fourier Transform altered by jittered sampling epochs », Proc. of I.C.A.S.S.P. 82, Intern. Conf. On Acoustic, Speech and Signal Processing, p 1837-1840, 1982.

[11] F. Cottet, M. Courtès, M. Holle, « Traitement de la gigue temporelle pour les ordonnancements par échéance », Proc. of RTS conferences, p. 63-77, Jan. 98.

[12] L. David, F. Cottet, E. Grolleau, « Maîtrise de la gigue temporelle avec les algorithmes d'ordonnancement DM et ED », 2000, Rapport interne 00 003 : LISI-ENSMA, Futuroscope, France.

[13] J.P. Babau, F. Cottet, « Méthodologie temporelle des applications temps réel à contraintes strictes » – JESA volume 32 – n°5,6 – Sept 98 – p 595.

[14] J. Blazewick, « Scheduling dependent tasks with different arrival times to meet deadlines », Modelling and Performance Evaluation of Computers Systems, North-Holland Publishing Company, 1976.

[15] H. Chetto, M. Silly, T. Bouchentouf, « Dynamic scheduling of real-time task under precedence constraints », Real-time Systems, 2, p181-194, 1990.

[16] Knuth, D.E., *Seminumerical algorithms*. 2nde Edition, Ed. The Art of Computer Programming. Vol. 2, 1981: Addison Wesley. 689 pages.